(2)

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE
90/11/02

3. REPORT TYPE AND DATES COVERED
Annual Technical Report
08/01/89 to 07/31/90

4. TITLE AND SUBTITLE
STABILITY and ADAPTATION of NEURAL NETWORKS

5. FUNDING NUMBERS
AFOSR-88-0236
Project No. 2305/B3
USC Acc't. No. 53-4513-0438

6. AUTHOR(S)
Professor Bart Kosko

7. PERFORMING ORGANIZAITON NAME(S) AND ADDRESS(ES)
University of Southern California
Department of Electrical Engineering-Systems/S.I.P.I
University Park/MC-0272
Los Angeles, CA 90089

8. PERFORMING ORGANIZATION REPORT NUMBER
AFOSR-TR- 90 1174

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Office of Scientific Research
AFOSR/PKD Attn.: Anne Sprunt
Building 410
Bolling AFB, D.C. 20332-6448

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFOSR-88-0236

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
UNCLASSIFIED/UNLIMITED

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

Focused on unsupervised learning and adaptive fuzzy systems. Explored the differential competitive learning (DCL) law. We successfully benchmarked DCL against su pervised competitive learning for phoneme recognition and centroid estimation. Proved structural stability for general competitive learning laws. Developed product-space clustering to develop adaptive fuzzy systems, which grow structured fuzzy rules from training data without supervision. Successfully benchmarked adaptive fuzzy systems against neural-network truck-and-trailer systems and Kalman-filter control systems for realtime target tracking.

14. SUBJECT TERMS

15. NUMBER OF PAGES
9 pages + copies of publications

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF REPORT	19. SECURITY CLASSIFICATION OF REPORT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UNLIMITED

ANNUAL REPORT

STABILITY AND ADAPTATION OF NEURAL NETWORKS

AFOSR-88-0236

Period Covered: August 1989 - July 1990

Bart Kosko, Principal Investigator

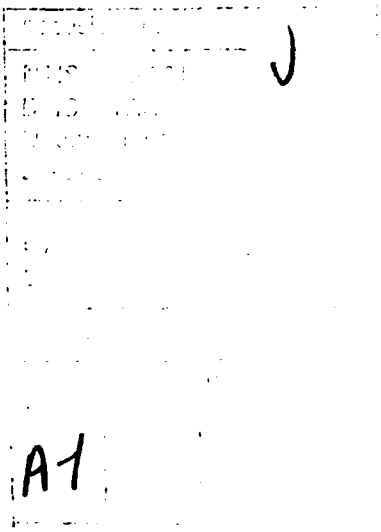
Department of Electrical Engineering--Systems
Signal and Image Processing Institute
University of Southern California
Los Angeles, California 90089-0272

ANNUAL REPORT: STABILITY AND ADAPTATION OF NEURAL NETWORKS

AFOSR-88-0236

Period Covered: August 1989 - July 1990

Principal Investigator: Bart Kosko, Ph.D.



1.0 RESEARCH SUMMARY

This year we focused on unsupervised learning and adaptive fuzzy systems. Graduate students Seong-Gon Kong and Peter Pacini assisted me in simulations and paper preparation. (USC's School of Engineering, not AFOSR, financially supported Peter Pacini.)

In unsupervised learning, we explored the differential competitive learning law. This unsupervised learning law differs from the standard competitive learning law, which modifies a synapse only if its post-synaptic neuron "wins" a competition for activation or pattern stimulation. The differential competitive learning (DCL) law modifies the synapse only if its post-synaptic neuron changes. Standard competitive learning ignores this

instantaneous win-rate information. The plus-or-minus sign of the neuronal time derivatives endows DCL with many of the coding properties shared by supervised competitive learning laws, which reward or punish synapses, according as the post-synaptic neuron classifies or misclassifies an input pattern, by changing the sign of a difference term.

We showed that DCL behaves as a type of adaptive delta-modulation procedure. When sampling highly correlated data, such as speech data, the pairwise difference of of samples provably contains more information (has less variance) than the samples themselves.

We tested DCL against both unsupervised and supervised competitive learning (SCL) for centroid estimation and for phoneme recogntion. DCL consistently outperformed SCL, which always outperformed unsupervised competitive learning. In the nonlinear case, DCL synaptic vectors converged to pattern-class centroids faster than SCL synaptic vectors conv rged to them. In the linear case, both types of synaptic vectors converged equally quickly to centroids. But once the synaptic vectors reached the centroids, DCL synaptic vectors wandered less about the centroids than the SCL synaptic vectors wandered. (Since we modeled learning with stochastic differential equations, stochastic equilibria corresponds to Brownian wandering about a fixed point.) The paper "Differential Competitive Learning for Centroid Estimation and Phoneme Recognition" summarizes these results and will appear in the January 1991 issue of the IEEE Transactions on Neural Networks.

In adaptive fuzzy systems, I developed the general AFAM (adaptive fuzzy associative memory) methodology and successfully applied it to two control problems: backing up a truck-and-trailer rig in a parking lot (a mathematically unsolved control problem) and realtime target tracking. This research, along with my pure research on fuzzy set theory, published this summer as "Fuzziness vs. Probability" in the International Journal of General Systems, has generated widespread technical and popular interest. Several popular and technical publications have featured it. These publications include Electronic Engineering Times, the Los Angeles Times, Popular Science, the Economist, and Breakthroughs. As program chairman of the first international conference on neural network and fuzzy theory in Iizuka Japan in July 1990, I presented these concepts to a wide international audience in my plenary lecture.

The key research contribution was product space clustering (PSC). I developed the geometry of fuzzy systems as mappings between unit hypercubes. (Fuzzy sets define points in unit hypercubes. Nonfuzzy sets define the 2^n vertices of an n-dimensional hypercube.) PSC estimates a fuzzy system as a surface in the system's input-output product state space. We partition the input-output state space into FAM cells. Each FAM cell defines a fuzzy "rule" or general association of fuzzy output descriptions with fuzzy input descriptions: If the traffic is HEAVY in one direction, then keep the light green LONGER in that direction. (HEAVY, LONGER) defines a FAM rule in

the input-output state space as region or mini-Cartesian product. PSC estimates these regions with unsupervised learning.

I showed how to use differential competitive learning to adaptively, quickly, and reliably generate banks of structured fuzzy rules given only the input-output training data generated by the physical process, the human controller, or, in general, the system we wish to estimate. We can present the same input-output data to a neural network. The neural network may or may not accurately estimate the underlying unknown function (or joint probability distribution). But it can generate only a black-box or "model-free" estimate. Stand-alone neural networks do not generate stuctured rules.

We benchmarked neural and fuzzy systems for backing up a truck, and also a truck-and-trailer, in a parking lot. Both systems controlled the truck and truck-and-trailer successfully. But the fuzzy system was orders of magnitude easier to construct and, in the adaptive case, train. We could also modify the fuzzy system directly by manipulating its bank of FAM rules. We tested the fuzzy system's robustness by removing random subsets of FAM rules and by deliberately important rules with destructive or "sabotage" rules. In general we found that fuzzy system performance degrades significantly only if we remove over 50% of the FAM rules. We also showed how to convert <u>every</u> neural network system to a structured fuzzy system, complete with bank of FAM rules, that approximates the underlying neural system. We demonstrated this for both the backpropagation truck and truck-

and-trailer systems. The generated fuzzy systems performed comparably to the original fuzzy systems.

We benchmarked an adaptive fuzzy system for realtime target tracking against an "optimal" linear Kalman-filter control system. Again both systems controlled the process well. The fuzzy system gave finer control, involved far less computation, required no assumption of how control outputs mathematically depended on control inputs, and proved robust when we removed FAM rules or replaced them with sabotage rules. The Kalman-filter controller proved sensitive to the variance of the unmodeled-effects parameter.

2.0 PUBLICATIONS

The above research led to several technical papers. We published some in proceedings and some in journals. Others await appearance or remain in the review process. This report includes copies of papers.

2.1 Journal Papers

1. Kosko, B., "Unsupervised Learning in Noise," IEEE

Transactions on Neural Networks, vol. 1, no. 1, 44 - 57, March 1990.

2. Kosko, B., "Structural Stability of Unsupervised Learning in Feedback Networks," IEEE Transactions on Automatic Control, in press, 1990.

3. Kosko, B., "Fuzziness vs. Probability," International Journal of General Systems, vol. 17, no. 2, 211 - 240, 1990.

4. Kosko, B., "Differential Competitive Learning for Centroid Estimation and Phoneme Recognition," with S.G. Kong, IEEE Transactions on Neural Networks, to appear, January 1991.

5. Kosko, B., "Stochastic Competitive Learning," in review at IEEE Transactions on Neural Networks, 1990.

6. Kosko, B., "Adaptive Fuzzy Systems for Backing Up a Truck-and-Trailer," with S.G. Kong, in review at IEEE Transactions on Neural Networks, 1990.

7. Kosko, B., "Adaptive Fuzzy System for Target Tracking," with P.J. Pacini, in review at IEEE Transactions on Automatic Control, 1990.

8. Kosko, B., "Fuzzy Associative Memories," to be submitted to Neural Networks, 1990.

2.2 Conference Proceedings Papers

1. Kosko, B., "Stochastic Competitive Learning," Proc. IJCNN-90, vol. II, 215 - 226, June 1990.

2. Kosko, B., "Comparison of Fuzzy and Neural Truck Backer-Upper Control Systems," with S.G. Kong, Proc. IJCNN-90, vol. III, 349-358, June 1990.

3. "Differential Competitive Learning for Centroid Estimation and Phoneme Recognition," Proc. European Conference on neural Netowrks, Prague, Czechoslovakia, September 1990.

3.0 NEXT-YEAR RESEARCH OBJECTIVES

In the third year of this research program we will continue to explore the relationship between unsupervised neural network systems and fuzzy systems.

We need to explore both the feedback dynamics of differential competitive learning and the feedforward encoding structure of pulse-coded DCL synapses. We have not fully exploited the delta-

modulation nature of DCL. The delta-modulation structure of DCL suggests that digital DCL systems may provide viable highspeed communication devices.

We need to explore the adaptive fuzzy methodology outside the domain of control. Two promising areas are image/signal processing and communication theory. The AFAM methodology may allow us to estimate image-compression schemes without detailed eigenvector math models. Communication systems depend on local highspeed decisions made with uncertain, usually probabilistic, information. Adaptive fuzzy systems may allow us to introduce "intelligent communication" at modulation, spreading/depsreading, or encoding/decoding level.

Unsupervised Learning in Noise

BART KOSKO, MEMBER, IEEE

Abstract—The structural stability of real-time unsupervised learning in feedback dynamical systems is demonstrated with the stochastic calculus. Structural stability allows globally stable feedback systems to be perturbed without changing their qualitative equilibrium behavior. These stochastic dynamical systems are called *random adaptive bidirectional associative memory* (RABAM) models, which include several popular nonadaptive and adaptive feedback models, such as the Hopfield circuit and the ART-2 model. RABAM networks can adapt with different stable unsupervised learning laws. These include the signal Hebb, competitive, and differential Hebb laws. A new hybrid learning law, the *differential competitive law*, which uses the neuronal signal velocity as a local unsupervised reinforcement mechanism, is introduced and its coding and stability behavior in feedforward and feedback networks is examined. This analysis is facilitated by the recent Gluck-Parker pulse-coding interpretation of signal functions in differential Hebbian learning systems. The second-order behavior of RABAM Brownian-diffusion systems is summarized by the RABAM noise suppression theorem: The mean-squared activation and synaptic velocities decrease exponentially quickly to their lower bounds, the instantaneous noise variances driving the system. This result is extended to the RABAM annealing model, which provides a unified framework from which to analyze Geman-Hwang combinatorial optimization dynamical systems and continuous Boltzmann machine learning.

I. Structural Stability in Hardware, Biology, and Manifolds

HOW robust are unsupervised learning systems? What happens if real-time synaptic mechanisms are perturbed in real time? Will shaking disturb or prevent equilibria? What effect will thermal noise processes, electromagnetic interactions, and component malfunctions have on large-scale implementations of unsupervised neural networks? How biologically accurate are unsupervised neural models that do not model the myriad electrochemical, molecular, and other processes found at synaptic junctions and membrane potential sites?

These questions are different ways of asking a more general question: is unsupervised learning *structurally stable*? Structural stability [9], [42] allows globally stable feedback systems to be perturbed without changing their qualitative equilibrium behavior. This increases the reliability of large-scale hardware implementations of such networks. It also increases their biological plausibility, since the myriad synaptic and neuronal processes missing from neural network models now are modeled, but as net random unmodeled effects that do not affect the structure of the global network computations.

Structural stability differs from the global stability, or convergence to fixed points, that endows some feedback networks with content addressable memory, and other, computational properties. Globally stable systems can be sensitive to initial conditions. Different inputs states can converge to different limit states; else memory capacity is trivial. Structural stability is insensitivity to small perturbations. Such perturbation preserves qualitative properties. In particular, basins of attractions maintain their basic shape. In some intuitive sense, chaos [36] is the antithesis of structural stability, or, more accurately, structurally stable fixed-point attractors (since chaotic attractors can be structurally stable).

The formal approach to structural stability uses the transversality techniques of differential topology [17], the study of global properties of differentiable manifolds. Manifolds A and B have nonempty transversal intersection in R^n if the tangent spaces of A and B span R^n at every point of intersection, if locally the intersection looks like R^n. Two lines intersect transversely in the plane but not in 3-space, 4-space, or higher n-space. If the lines are shaken in 2-space, they still intersect. If shaken in 3-space, the lines may no longer intersect. In Fig. 1, manifolds A and B intersect transversely in the plane at points a and b. Manifolds B and C do not intersect transversely at c.

An indirect approach to structural stability uses the calculus of stochastic differential and integral equations [35], [41]. This is the approach used in this paper. The stochastic-calculus approach abstracts statistically relevant behavior from large sets of functions. The differential-topological approach, in contrast, is concerned with all possible behavior of all functions (open dense sets of functions). This makes the analysis extremely abstract and calculations cumbersome and often impractical.

The stochastic calculus is difficult to work with as well, but usually less difficult than transversality techniques. The new complexity that arises in passing from systems of differential equations to systems of stochastic differential equations is due to the nature of solution points. In algebraic equations, such as $2x + 3 = 4x$, points in the solution space are numbers. Solutions to differential equations are functions. Solutions to stochastic differential equations are random processes [41].

Manuscript received April 10, 1989; revised October 9, 1989. This work was supported by the Air Force Office of Scientific Research (AFOSR 88-0236). An earlier version of this paper was presented at the 1989 International Joint Conference on Neural Networks (IJCNN-89), Washington, DC, June 18–22, 1989.

The author is with the Department of Electrical Engineering, Systems, Signal, and Image Processing Institute, University of Southern California, Los Angeles, CA 90089-0272.

IEEE Log Number 8933088.

1045-9227/90/0300-0044$01.00 © 1990 IEEE

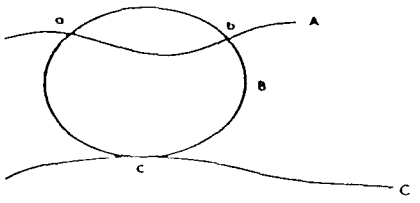


Fig. 1. Manifold intersection in the plane (manifold R^2). Intersection points a and b are transversal. Point c is not: manifolds B and C need not intersect if even slightly perturbed. No points are transversal in 3-space unless B is a sphere.

Below we demonstrate the structural stability of many types of unsupervised learning in the stochastic sense. The key idea is to use the scalar-valued Lyapunov function of globally stable feedback networks but in a random framework. Then the old Lyapunov function is a random variable at each moment of time t, so it cannot be minimized as when it was a scalar at each t. The trick is to minimize its expectation, its average value, which is a scalar at t.

II. Four Unsupervised Associative Learning Laws

The distinction between supervised and unsupervised learning depends on information. In pattern-recognition theory, for instance, the distinction is in terms of knowledge of class boundaries. Pattern recognition is supervised if the training algorithm requires knowing the class membership of the training samples, unsupervised if it does not require it.

A similar distinction holds in neural networks. Supervised learning invariably refers to deliberate gradient descent in the space of all possible synaptic values. Class membership information is needed to compute the numerical error vector or error signal that guides the gradient descent.

Unsupervised learning usually refers to the modification of biological synapses with physically local signal information. Class membership information of training samples is not needed. These systems adaptively cluster patterns into classes by, for example, evolving "winning" neurons in a competition for activation, or by evolving different basins of attraction in the state space. We shall restrict our attention to such biologically motivated learning methods, knowing that other types of unsupervised learning are possible and may be of practical engineering value.

Unsupervised learning laws are first-order differential equations that describe how synapses evolve in time with *locally available* information. This information usually involves synaptic properties or neuronal signal properties. In principle, and in mammalian brains or optoelectronic integrated circuits, other types of information may be locally available for computation, glial cells, specific and nonspecific hormones, background electromagnetic effects, or light pulses. These phenomena are modeled below as net random parameters. For the moment they will be ignored. Locality allows asynchronous synapses to operate in real time. Mathematically, it also greatly shrinks the function space of possible unsupervised learning laws.

Associativity further shrinks the function space. Globally, neural networks associate patterns with patterns. They estimate continuous functions. Locally, synapses are required to associate signals with signals. This leads to conjunctive, or multiplicative, learning laws constrained by locality. This in turn leads to at least three types of learning laws and a new hybrid law.

The four unsupervised associative learning laws discussed in this section are 1) the signal Hebb learning law, 2) the competitive learning law, 3) the differential Hebb learning law, and 4) the new hybrid law, the differential competitive learning law.

A. Signal Hebbian Learning

The *signal Hebb* learning law correlates neuronal signals, not activations:

$$\dot{m}_{ij} = -m_{ij} + S_i^X(x_i)\, S_j^Y(y_j) \tag{1}$$

where the overdot indicates time differentiation, m_{ij} is the synaptic efficacy of the directed axonal edge from the ith neuron in field F_X to the jth neuron in field F_Y, x_i and y_j are the respective real-valued activations or membrane potentials of the connected neurons, and S_i^X and S_j^Y, hereafter abbreviated to S_i and S_j, are the bounded monotone-nondecreasing signal functions of the connected neurons that transduce their time-averaged potential differences into time-averaged frequencies of pulse trains, and where, as in all equations in this paper, scaling constants can be multiplied or added where desired. The logistic signal function $S(x) = (1 + e^{-cx})^{-1}$, with $c > 0$, remains the most popular signal function for simulations and applications. The logistic signal function is also strictly monotone increasing, since $S' = dS(x)/dx = cS(1 - S) > 0$. Strict monotonicity strengthens stability results.

The solution to (1) is an integral equation since in general x_i and y_j depend on m_{ij}. The key component of this integral equation is an exponentially weighted average of sampled patterns:

$$m_{ij}(t) = m_{ij}(0)e^{-t} + \int_0^t S_i(s)\, S_j(s) e^{s-t}\, ds. \tag{2}$$

The exponential weight is inherent in the first-order structure of (1). It produces a *recency effect* on memory, as in our everyday exponential decrease in retained information. This well-known recency effect is the thrust of philosopher David Hume's quote: "The liveliest thought still is inferior to the dullest sensation." Nothing is more vivid than now.

B. Competitive Learning

The *competitive learning law* is obtained from (1) if the passive decay term $-m_{ij}$ is modulated by the appropriate local signal:

$$\dot{m}_{ij} = S_j[S_i - m_{ij}]. \tag{3}$$

The "competitiveness" in (3) is indirect. The assumption is that neurons compete for activation in the field F_Y

in the sense that the symmetric (distance-dependent) intrafield connections of F_Y are laterally inhibitive: the square symmetric matrix Q of intrafield connections is positive main-diagonal and nonpositive off-diagonal, or, more generally, Q has nonnegative blocks on its main diagonal and nonpositive blocks elsewhere. Then S_j is a win-loss index of the jth F_Y neuron's performance. In practice [39] S_j is invariably a 0-1 threshold function or steep logistic function, which behaves as a threshold function. Then (3) says *learn only if win*. If the jth unit wins, the signal pattern $S(X) = (S_1(x_1), \ldots, S_n(x_n))$ generated at F_X is encoded as the jth column of the n-by-p connection matrix exponentially quickly. This "grandmother synapse" effect differs from Hebbian learning, where pattern information is superimposed on all of M. Then every synapse participates in learning new patterns while, unfortunately, forgetting learned patterns.

Both (1) and (2) were studied as early as the 1960's by Grossberg [12]. Kohonen [24] and Hecht-Nielsen [15] use the competitive law (3) statistically for unsupervised clustering in their respective self-organizing map and counterpropagation networks. The p columns of M then tend toward the centroids of the sampled p decision classes, even though the underlying probability density functions are unknown.

C. Differential Hebbian Learning

The *differential Hebb* law [25]–[27], [32], [33], and its variants, correlates signal velocities as well as signals:

$$\dot{m}_{ij} = -m_{ij} + S_i S_j + \dot{S}_i \dot{S}_j \tag{4}$$

where, by the chain rule

$$\frac{dS_i(x_i)}{dt} = \frac{dS_i}{dx_i}\frac{dx_i}{dt} = S_i' \dot{x}_i.$$

If signals are locally available to synapses, so are signal velocities, at least implicitly. Since the signal function S_i is an abstraction of time-averaged spiking frequencies, S_i is often assumed nonegative. Then Hebbian synapses (1) can only grow in time. Signal velocities, of course, can be both positive and negative. Correlated (lagged) signals provide a local "arrow of time" that synapses can exploit [33] to encode time-varying patterns as limit cycles. Klopf [21]–[23] independently arived at a similar discrete (difference) version of (4) in his drive-reinforcement theory of animal learning.

Recently Gluck and Parker [10], [11] showed that differential Hebbian learning becomes significantly more plausible in nervous systems if we recall that real neurons transmit discrete pulse-coded information and we structure the signal functions S_i and S_j accordingly. Suppose x_i and y_j are pulse functions: $x_i(t) = 1$ if a pulse occurs at time t, 0 if not, and similarly for $y_j(t)$. Then the signal frequencies S_i and S_j can be estimated as exponentially weighted time averages:

$$S_i(t) = \int_{-\infty}^{t} x_i(s) e^{s-t}\, ds \tag{5}$$

$$S_j(t) = \int_{-\infty}^{t} y_j(s) e^{s-t}\, ds. \tag{6}$$

By recalling the form of the solution to a linear inhomogeneous, first-order differential question, the signal velocities are seen to be simple, locally available, differences:

$$\dot{S}_i(t) = x_i(t) - S_i(t) \tag{7}$$

$$\dot{S}_j(t) = y_j(t) - S_j(t). \tag{8}$$

Thus a signal velocity has the form of a reinforcement signal: a pulse less the current expected frequency of pulses. As Gluck and Parker observe, not only are these differences locally available, they can be computed in real time without unstable differencing techniques.

For stability purposes, we note another consequence of pulse-coded signal functions. They show how Hebbian learning can be a special case of differential Hebbian learning. Suppose the Hebb product $S_i S_j$ in (4) is scaled down to zero:

$$\dot{m}_{ij} = -m_{ij} + \dot{S}_i \dot{S}_j. \tag{9}$$

This is the "classical" differential Hebb law [25]–[27]. Then substituting (7) and (8) into (9) gives

$$\dot{m}_{ij} = -m_{ij} + S_i S_j + [x_i y_j - x_i S_j - y_j S_i] \tag{10}$$

which is equivalent to the signal Hebb law (1) if and only if the term in braces is zero. Thus the simple differential Hebb law (9), and of course (4) suitably scaled, reduces to the signal Hebb law when no pulses occur, when $x_i(t) = y_j(t) = 0$. This happens frequently. For, in any connected time interval, the set v of times where pulses occur, $\{t': x_i(t') = 1\}$, has Lebesgue measure zero. (Consider pulses at rational time points or at Cantor set points.) This interpretation, though, would imply [38] by (5) and (6) that $S_i = S_j = 0$ almost everywhere, so the integrals in (5) and (6) would have to be replaced with discrete sums (using point-mass measures).

The infrequency of unit pulses occurs while the synapse m_{ij} continually modifies its behavior. When instantaneous pulse information is not available, the synapse "fills in" with expected pulse frequencies, and hence Hebbian learning. Since signal Hebbian learning is unconditionally stable (the ABAM theorem, reviewed below) in many nonlinear dynamical systems, including popular feedback neural networks, pulse-coded differential Hebbian dynamical systems may be stable over a wider range of system parameters than earlier velocity-acceleration stability assumptions [32], [33] suggested.

D. Differential Competitive Learning

The fourth unsupervised learning law is a new hybrid learning law, the *differential competitive* law:

$$\dot{m}_{ij} = \dot{S}_j [S_i - m_{ij}]. \tag{11}$$

The idea is *learn only if change*. As with the competitive learning law (3), the neurons in F_Y compete for acti-

vation, and the nonnegative signal functions S_j keep score. The signal velocity $\dot{S}_j$ in (11) is a local *reinforcement* mechanism. Its sign indicates whether the jth neurons are winning or losing, and its magnitude measures by how much. The coding and dynamical behavior of (11) can be analyzed with the pulse-coding interpretation [10], [11] of signal functions and by comparison with Kohonen's recent "supervised" adaptive-vector-quantization algorithm [24].

The pulse-coded differential competitive learning law is the difference of nondifferential competitive laws:

$$\dot{m}_{ij} = (y_j - S_j)[S_i - m_{ij}] \tag{12}$$

$$= y_j[S_i - m_{ij}] - S_j[S_i - m_{ij}] \tag{13}$$

where x_i is a 0-1 pulse function. Hence the standard competitive learning law (3) is recovered when $y_j = 1$ and $S_j = 0$. This occurs when the jth unit has just won the competition for activation within F_Y.

Usually in a competition there are many more losers than winners. So suppose the jth neuron in F_Y is a loser at time t. Then $y_j(t) = 0$ holds and has held over some, perhaps short, past interval $[t', t]$. Then $S_j(t) = 0$ (or nearly 0) by the exponential-weight structure of (6). So no change, no learning.

Now suppose the jth unit wins in the next instant t. Then $y_j = 1$ over some interval $[t, t'']$ of nonzero Lebesgue measure. During this interval the exponential-weight structure of S_j soon drives S_j toward 1, which we take as the upper bound of S_j. This means m_{ij} quickly approaches a positively scaled version of the signal S_i.

Now suppose the jth unit goes from winning to losing. Then at first $y_j = 0$ and $S_j = 1$. As S_j quickly falls to zero, learning slows then stops when $y_j = S_j = 0$. Meanwhile m_{ij} has "moved away" from the signal S_i. The signal velocity $\dot{S}_j$ has "punished" the jth unit.

Kohonen [24] uses a sign change to punish misclassifying prototype vectors trained with the competitive learning law in his feedforward "supervised" adaptive vector quantization (AVQ) system. In vector formulation, the p reference vectors $m_1(t), \ldots, m_p(t)$ are the respective prototypes at time t of the p decision classes $D_1, \ldots, D_p$ that partition the signal space R^n. The p reference vectors are also the p columns of the synaptic matrix M. $m_i = (m_{i1}, \ldots, m_{in})$ is the fan-in of synapses of the ith neuron in F_Y. All F_Y neurons are engaged in winner-take-all competition. Given a random training sample vector $x(t)$ presented at F_X, the F_Y competition is summarized by finding the reference vector $m_j(t)$ closest to $x(t)$ in Euclidean distance: $\|x - m_j\| = \min\{\|x - m_i\|: i = 1, \ldots, p\}$. "Supervision" means we know which decision class the random vector x was chosen from. If x belongs to D_j, the class represented by m_j, then m_j is rewarded by moving m_j a little closer to x. This allows m_j to gradually approximate the centroid of D_j. (The centroid, or conditional expectation, minimizes the mean-squared-error of vector quantization [37].) Else if x does not belong to D_j, m_j is punished for misclassifying x as a D_j pattern by moving m_j a little farther away from x, presumably out of regions of misclassification. This is achieved by a simple sign change:

$$m_j(t+1) = \begin{cases} m_j(t) + c(t)[x(t) - m_j(t)], & x \in D_j \\ & (14) \\ m_j(t) - c(t)[x(t) - m_j(t)], & x \notin D_j \\ & (15) \end{cases}$$

$$m_i(t+1) = m_i(t) \text{ for all losing neurons in } F_Y \tag{16}$$

where $c(0), c(1), c(2), \ldots$ is a slowly decreasing sequence of small ($c(0) < 1$) learning constants. Kohonen's "unsupervised" AVQ algorithm eliminates the punishment equation (15) and relaxes (14) by allowing $x(t)$ to belong to any decision class. The unsupervised algorithm is clearly a discrete stochastic version of the competitive law (3) in vector notation. Kohonen shows that under appropriate statistical conditions, the equilibrium condition of the AVQ unsupervised-clustering algorithm occurs when the p reference vectors m_i asymptotically arrive at the centroids of their respective decision classes. Kohonen next shows that the equilibrium condition of the supervised AVQ algorithm is similar in structure to that of the optimum unit-cost Bayes classifier, and cites simulation data in support of this similarity.

The differential competitive law (11) can be viewed as a local unsupervised *approximation* of Kohonen's supervised AVQ algorithm. Indeed preliminary simulations of (11) in stochastic feedforward mode show similar classification performance in many noise environments.

The pulse-coded differential competitive law (12), as discussed above, can be expected to often behave as the competitive law (3) with 0-1 threshold signal function S_j. This is precisely when the competitive law has been shown [32] globally stable when embedded in the nonlinear dynamical systems below. For this reason, we here limit the stability analysis of the differential competitive law to that of the competitive law with steep signal function S_j. We similarly limit the stability analysis of the differential Hebb law (4) to the analysis of the signal Hebb law, even though differential Hebb dynamical systems are known [32], [33] globally stable in the special case that signal velocities are comparable to signal accelerations.

III. Unidirectional and Bidirectional Nonlinear Dynamical Systems

We study nonlinear dynamical systems described by Cohen-Grossberg [6], [14] dynamics. In the unidirectional or autoassociative case, when $F_X = F_Y$ and $M = M^T$, a neural network possesses Cohen-Grossberg dynamics if its activation equations can be written in the abstract form

$$\dot{x}_i = -a_i(x_i)\left[b_i(x_i) - \sum_j S_j(x_j)m_{ij}\right] \tag{17}$$

where $a_i(x_i) \geq 0$ is an amplification function, b_i is arbitrary so long as it keeps the integrals bounded in the Lyapunov functions below, and S_i is a bounded monotone nondecreasing ($S_i' \geq 0$) signal function. The global stability of nonlearning autoassociative systems described by (17) is ensured by the Cohen-Grossberg theorem [6], which is abstractly equivalent—in the sense that $R^n \times R^n = R^{n+n}$—to the BAM theorem below for nonlearning heteroassociative networks and a special case of the ABAM theorem reviewed in the next section.

Perhaps the most important special cases of (17) are additive and shunting networks, the popular versions of which are the respective Hopfield circuit [19] and the Hodgkin-Huxley membrane equation [18]. Grossberg [14], has also shown that (17) reduces to the additive brain-state-in-a-box model of Anderson [1], [2] and the shunting masking field model [7] upon appropriate change of variables. An autoassociative system has *additive* activation dynamics if the amplification function a_i is constant and the b_i function is linear. For instance, if $a_i = 1/C_i$, $b_i = (x_i/R_i) - I_i$, $S_i(x_i) = g_i(x_i) = V_i$, and constant $m_{ij} = m_{ji} = T_{ij} = T_{ji}$, where C_i and R_i are positive constants and input I_i is constant or slowly varying relative to fluctuations in x_i, then (17) reduces to the Hopfield circuit [19]:

$$C_i\dot{x}_i = -\frac{x_i}{R_i} + \sum_j V_j T_{ij} + I_i. \tag{18}$$

Grossberg [13] has shown that neurons with additive dynamics saturate at their upper bounds (if they have them) when inputs are arbitrarily large, thus ignoring the relative pattern information in the input pattern $(I_1, \ldots, I_n)$.

An autoassociative network has *shunting* or multiplicative activation dynamics when the amplification function a_i is linear and b_i is nonlinear. For instance, if $a_i = -x_i$, $m_{ii} = 1$ (self-excitation in lateral inhibition), and $b_i = (1/x_i)[-A_i x_i + B_i(S_i + I_i^+) - x_i(S_i + I_i^+) - C_i(\Sigma_{j \neq i} S_j m_{ij} + I_i^-)]$, gives the distance-dependent ($m_{ij} = m_{ji}$) unidirectional shunting network:

$$\dot{x}_i = -A_i x_i + (B_i - x_i)[S_i(x_i) + I_i^+] - (C_i + x_i)\left[\sum_{j \neq i} S_j(x_j) m_{ij} + I_i^-\right] \tag{19}$$

where A_i is a positive decay constant and B_i and C_i are positive saturation constants. The first term on the right-hand side of (19) is a passive decay term. The second and third terms are, respectively, positive and negative feedback terms. (Strictly speaking, $a_i(x_i)$ must be kept positive. x_i can always be translated to achieve this.) If the shunting x_i terms in the positive and negative feedback terms are scaled to zero, (19) reduces to an additive model. Grossberg also showed that shunting models do not saturate when presented with arbitrarily large positive inputs. They remain sensitive to the relative pattern information in $(I_1, \ldots, I_n)$. Perhaps more important for neurobiologists, Grossberg [13], [14] observed that the shunting model (19) is naturally generalized by the celebrated Hodgkin-Huxley membrane equation:

$$c\frac{\partial V_i}{\partial t} = (V^p - V_i)g_i^p + (V^+ - V_i)g_i^+ + (V^- - V_i)g_i^- \tag{20}$$

where V^p, V^+, and V^- are respective passive, excitatory (sodium Na^+), and inhibitory (potassium K^+) saturation upper bounds with corresponding shunting conductances g_i^p, g_i^+, and g_i^-, and where the constant capacitance $c > 0$ scales time. The shunting model (19) becomes the membrane equation (20) if $V_i = x_i$, $V^p = 0$, $V^+ = B_i$, $V^- = -C_i$, $g_i^p = A_i$, $g_i^+ = S_i(x_i) + I_i^+$, and $g_i^- = \Sigma_{j \neq i} S_j m_{ij} + I_i^-$.

Continuous *bidirectional associative memories* [28]–[32] (BAM's) arise when two (or more) neural fields F_X and F_Y are connected in the forward direction, from F_X to F_Y, by an arbitrary n-by-p synaptic matrix M and connected in the backward direction, from F_Y to F_X, by the p-by-n matrix $N = M^T$, where M^T is the transpose of M. BAM activations also possess Cohen–Grossberg dynamics, and their extensions:

$$\dot{x}_i = -a_i(x_i)\left[b_i(x_i) - \sum_{j=1}^{p} S_j(y_j) m_{ij}\right] \tag{21}$$

$$\dot{y}_j = -a_j(y_j)\left[b_j(y_j) - \sum_{i=1}^{n} S_i(x_i) m_{ij}\right] \tag{22}$$

with corresponding Lyapunov function L:

$$L = -\sum_i \sum_j S_i S_j m_{ij} + \sum_i \int_0^{x_i} S_i'(\theta_i)\, b_i(\theta_i)\, d\theta_i + \sum_j \int_0^{y_j} S_j'(\epsilon_j)\, b_j(\epsilon_j)\, d\epsilon_j$$

where the functions b_i and b_j must be suitably constrained to keep L bounded.

The quadratic form in L is bounded because the signal functions S_i and S_j are bounded. Boundedness of the integral terms requires additional technical hypotheses to avoid pathologies as discussed by Cohen and Grossberg [6]. For our purpose we simply assume the integral terms are bounded.

All BAM results extend to any number of BAM-connected fields. Complex topologies are possible and, in theory, will equilibrate as rapidly as the two-layer BAM system. The back-and forth flow of information in a BAM facilitates natural large-scale optical implementations [20], [28].

The BAM model (21), (22) clearly reduces to the Cohen-Grossberg model if both neural fields collapse into one, $F_X = F_Y$, and the constant matrix M is symmetric ($M = M^T$). Conversely, the BAM system, which is always globally stable, can be abstractly viewed [30] as symmetrizing an arbitrary matrix M. For if the two BAM fields

are abstractly concatenated into a new field F_Z, $F_Z = F_X \cup F_Y$, with zero block diagonal synaptic matrix W that contains M and M^T as respective upper and lower blocks, then the BAM dynamical system (21), (22) is equivalent to the autoassociative system (17).

The BAM system (21) includes additive and shunting models. If $a_i = 1 = a_j$, $b_i = x_i - I_i$, and $b_j = y_j - J_j$, for relatively constant inputs I_i and J_j, then an *additive* BAM [30], [31] results:

$$\dot{x}_i = -x_i + \sum_j S_j(y_j)m_{ij} + I_i \tag{23}$$

$$\dot{y}_j = -y_j + \sum_i S_i(x_i)m_{ij} + J_j \tag{24}$$

where again constants can be added or multiplied as desired. More generally, if $a_i = -x_i$, $a_j = -y_j$, $b_i = (1/x_i)[-x_i + (B_i - x_i)[S_i(x_i) + I_i^+] - x_i I_i^-]$, and $b_j = (1/y_j)[-y_j + (B_j - y_j)[S_j(y_j) + J_j^+] - y_j J_j^-]$, then a *shunting* BAM [30] results:

$$\dot{x}_i = -x_i + (B_i - x_i)[S_i + I_i^+] - x_i\left[\sum_j S_j m_{ij} + I_i^-\right] \tag{25}$$

$$\dot{y}_j = -y_j + (B_j - y_j)[S_j + J_j^+] - y_j\left[\sum_i S_i m_{ij} + J_j^-\right]. \tag{26}$$

The shunting BAM (25), (26) reminds us that in general distance-dependent competition occurs within fields F_X and F_Y. Suppose the n-by-n matrix R and the p-by-p matrix S describe the distance-dependent ($R = R^T$, $S = S^T$) lateral inhibition within F_X and F_Y, respectively. Then the general BAM model (21), (22) must be augmented to a competitive BAM [29]:

$$\dot{x}_i = -a_i(x_i)\left[b_i(x_i) - \sum_j^p S_j(y_j)m_{ij} - \sum_k^n S_k(x_k)r_{ki}\right] \tag{27}$$

$$\dot{y}_j = -a_j(y_j)\left[b_j(y_j) - \sum_i^n S_i(x_i)m_{ij} - \sum_l^p S_l(y_l)s_{lj}\right]. \tag{28}$$

An *adaptive* bidirectional associative memory (ABAM) is a globally stable dynamical system with activation dynamics described by (21), (22) or (27), (28) and synaptic dynamics described by a first-order learning law. The original ABAM [30] restricted the choice of learning law to the signal Hebb law (1). Signal Hebb ABAM's are unconditionally globally stable, though limited in their ability to estimate continuous functions. Better, though more costly, estimation can be gotten with higher order signal Hebb ABAM's. For example, in autoassociative notation, the second-order signal Hebb ABAM [32] is described by (29)-(31):

$$\dot{x}_i = -a_i(x_i)\left[b_i(x_i) - \sum_j S_j(x_j)m_{ij} - \sum_j \sum_k S_j(x_j)\,S_k(x_k)n_{ijk}\right] \tag{29}$$

$$\dot{m}_{ij} = -m_{ij} + S_i(x_i)\,S_j(x_j) \tag{30}$$

$$\dot{n}_{ijk} = -n_{ijk} + S_i(x_i)\,S_j(x_j)\,S_k(x_k) \tag{31}$$

with corresponding Lyapunov function L:

$$L = -\tfrac{1}{2}\sum_i \sum_j S_i S_j m_{ij} - \tfrac{1}{3}\sum_i \sum_j \sum_k S_i S_j S_k n_{ijk} + \sum_i \int_0^{x_i} S_i'(\theta_i)\,b_i(\theta_i)\,d\theta_i + \tfrac{1}{4}\sum_i \sum_j m_{ij}^2 + \tfrac{1}{6}\sum_i \sum_j \sum_k n_{ijk}^2. \tag{32}$$

The Lyapunov function remains bounded in the adaptive case. The new terms

$$\tfrac{1}{4}\sum_i \sum_j m_{ij}^2 \quad \text{and} \quad \tfrac{1}{6}\sum_i \sum_j \sum_k n_{ijk}^2 \tag{33}$$

in (32) are bounded because the solutions to (30) and (31) are bounded since, ultimately, the signal functions S_i are bounded.

If $a_i(x_i) > 0$ and $S_i' > 0$, and if (32) is differentiated with respect to time, rearranged, and (29), (30) are used to eliminate terms, then L strictly decreases along trajectories, yielding asymptotic stability (and in general exponential convergence), since

$$\dot{L} = -\sum_i \frac{S_i'(x_i)}{a_i(x_i)}\dot{x}_i^2 - \frac{1}{2}\sum_i \sum_j \dot{m}_{ij}^2 - \frac{1}{3}\sum_i \sum_j \sum_k \dot{n}_{ijk}^2 < 0 \tag{34}$$

if any activation or synaptic velocity is nonzero. The strict monotonicity assumption $S_i' > 0$ and (33) further imply that $\dot{L} = 0$ if and only if all parameters stop changing: $\dot{x}_i = \dot{m}_{ij} = \dot{n}_{ijk} = 0$ for all i, j, k. All like higher order ABAM's are globally stable.

The restriction to signal Hebbian learning was relaxed [32] to allow competitive learning with (3) provided S_j is steep, and further relaxed to allow differential Hebbian learning with (4) provided signal velocities and signal accelerations agree in sign. A *competitive* ABAM (CABAM) results from (27), (28) if learning is governed by the competitive learning law (3) and if S_j behaves essentially as a 0-1 step function. For then, upon time differentiation, the appropriate Lyapunov function L takes the form

$$\dot{L} = -\sum_i \frac{S_i'(x_i)}{a_i}\dot{x}_i^2 - \sum_j \frac{S_j'(y_j)}{a_j}\dot{y}_j^2 - \sum_i \sum_j \dot{m}_{ij}[S_i(x_i)\,S_j(y_j) - m_{ij}]. \tag{35}$$

The trick is to eliminate $\dot{m}_{ij}$ in (34) with the competitive law (3) and exploit the 0-1 threshold (steep-sigmoid) behavior of S_j. Then the relevant product becomes nonnegative:

$$\dot{m}_{ij}[S_iS_j - m_{ij}] = S_j(S_i - m_{ij})[S_iS_j - m_{ij}]$$

$$= \begin{cases} 0, & S_j(y_j) = 0 \\ (S_i - m_{ij})^2, & S_j(y_j) = 1. \end{cases}$$

Thus both winners and losers in F_Y keep L decreasing and ensure that every CABAM is globally stable.

CABAM's are topologically equivalent to *adaptive resonance theory* (ART) systems [13]. The idea behind ART systems is *learn only if resonate*. Resonance, though, is simply joint stability at F_X and F_Y mediated by the forward connections M and the backward connections N. When $N = M^T$ and activation dynamics are described by (27), (28), ART models become CABAM models so long as learning is described by a globally stable learning law, in particular the competitive law (3) with steep signal function S_j. This is the case with the recent ART-2 model [5] since the activation (short-term memory) dynamics of F_X and F_Y are described by shunting equations and, in the notation of Carpenter and Grossberg, the learning (long-term memory) dynamics are described by CABAM-style competitive learning laws with threshold signal functions in F_Y:

$$\text{top-down } (F_Y \to F_X): \dot{z}_{ji} = g(y_j)[p_i - z_{ji}] \quad (36)$$

$$\text{bottom-up } (F_X \to F_Y): \dot{z}_{ij} = g(y_j)[p_i - z_{ij}] \quad (37)$$

where g is a threshold signal function and p_i is the signal pattern (itself involving complicated L^2-norm computations) transmitted from F_X. Equation (36) says matrix Z contains forward projections and its transpose Z^T contains backward connections.

In contrast, the earlier binary ART-1 model [4] is not *extended* by the CABAM model because Weber law structure is imposed on the forward "bottom-up" synaptic projections, and thus the forward and backward connection matrices are not related by transposition. This in part explains why binary inputs in ART-2 need not produce ART-1 behavior. It also suggests that the ART-2 model can in principle be similarly modified by adding Weber law structure to (36), producing an ART-2' model that is not a CABAM.

These connections among unsupervised feedback dynamical systems are summarized by the taxonomy in Fig. 2 of artificial neural networks (ANN's) and placed in context with unsupervised feedforward adaptive vector quantizers and the extremely popular supervised feedforward gradient-descent networks.

The more general RABAM model is developed below.

Finally, for completeness, we state the form of ABAM systems that adapt (and activate) with signal velocity information by using the differential Hebb learning law [33]:

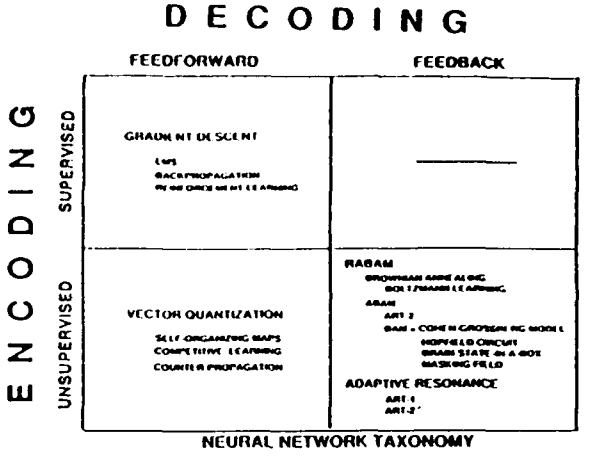


Fig. 2.

$$\dot{x}_i = -a_i(x_i)\left[b_i(x_i) - \sum_j S_j m_{ij} - \sum_j \dot{S}_j m_{ij}\right] \quad (38)$$

$$\dot{y}_j = -a_j(y_j)\left[b_j(y_j) - \sum_i S_i m_{ij} - \sum_i \dot{S}_i m_{ij}\right] \quad (39)$$

$$\dot{m}_{ij} = -m_{ij} + S_iS_j + \dot{S}_i\dot{S}_j \quad (40)$$

and the further assumptions $\dot{S}_i \approx \tilde{S}_i$, $\dot{S}_j \approx \tilde{S}_j$, where in general (40) can be loosened to only require that signal velocities and accelerations tend to have the same sign (as in clipped exponentials). The corresponding Lyapunov function now includes a "kinetic energy" term to account for signal velocities:

$$L = -\sum_i \sum_j S_iS_jm_{ij} - \sum_i \sum_j \dot{S}_i\dot{S}_jm_{ij}$$

$$+ \sum_i \int_0^{x_i} S_i'(\theta_i)\, b_i(\theta_i)\, d\theta_i$$

$$+ \sum_j \int_0^{y_j} S_j'(\epsilon_j)\, b_j(\epsilon_j)\, d\epsilon_j + \tfrac{1}{2} \sum_i \sum_j m_{ij}^2.$$

IV. Stability-Convergence Dilemma and the ABAM Theorem

Stability and convergence are equilibrium properties. *Stability* is equilibrium in a neuronal field: $(d/dt)F_X = 0$. *Convergence* is equilibrium in a synaptic web: $(d/dt)M = 0$. *Global stability* is joint stability and convergence for all inputs and all network parameters. *Pattern formation* occurs across field F_X when it stabilizes. The stable signals across F_X make up the formed pattern. Stability is trivial in a feedforward network.

Global stability is difficult to achieve in unsupervised feedback networks. After all, most feedback systems are unstable. Global stability requires a delicate dynamical balance between stability and convergence. Achieving such a balance is arguably the central problem in analyzing, and building, unsupervised feedback dynamical sys-

tems. The chief difficulty stems from the dynamical asymmetry between neural and synaptic fluctuations. Neurons fluctuate orders of magnitude faster than synapses: learning is slow. In real neural systems, neuronal fluctation may be at the millisecond level, while synaptic fluctuation may be at the second or even minute level.

The *stability-convergence dilemma* arises from the asymmetry in neuronal and synaptic fluctuation rates. The dilemma unfolds as follows. Neurons change faster than synapses change. Patterns form when neurons stabilize, when $(d/dt)F_X = 0$ and $(d/dt)F_Y = 0$. The slowly varying synapses M try to learn these patterns. Since the neurons are stable for more than a synaptic moment, the synapses begin to adapt to the neuronal patterns—learning begins. So $(d/dt)F_X = 0$ and $(d/dt)F_Y = 0$ imply $(d/dt)M \neq 0$. Since there are numerous feedback paths from the synapses to the neurons, the neurons tend to change state. So $(d/dt)M \neq 0$ implies $(d/dt)F_X \neq 0$ and $(d/dt)F_Y \neq 0$. *Learning tends to undo the very stability patterns to be encoded*, and hence the dilemma. In summary, for two fields of neurons F_X and F_Y connected in the forward direction by M and in the backward direction by M^T, the stability-convergence dilemma has four parts, described as follows.

A. Stability-Convergence Dilemma

1) *Asymmetry:* Neurons in F_X and F_Y fluctuate faster than the synapses M.

2) *Stability:* $\frac{d}{dt}F_X = 0$ and $\frac{d}{dt}F_Y = 0$ (pattern formation).

3) *Learning:* $\frac{d}{dt}F_X = 0$ and $\frac{d}{dt}F_Y = 0 \rightarrow \frac{d}{dt}M \neq 0$.

4) *Undoing:* $\frac{d}{dt}M \neq 0 \rightarrow \frac{d}{dt}F_X \neq 0$ and $\frac{d}{dt}F_Y \neq 0$.

The ABAM theorem [32] provides one resolution of the stability-convergence dilemma. The adaptive resonance concept provides another. Though as discussed in the previous section, the recent ART-2 instantiation of the concept is a CABAM. The ABAM theorem ensures the global stability, the joint stability and convergence, of dynamical systems with activation dynamics described by (21) and (22) and that learn according to the signal Hebb learning law (1). The extensions to competitive and differential Hebbian learning (and thus differential competitive learning) discussed above all require more assumptions than learning with the signal Hebb law, which requires none. Since the ABAM theorem is the starting point for the random-process extension to the RABAM theorem below, we review its statement and proof.

B. ABAM Theorem

Every signal Hebb BAM is asymptotically stable, where the network dynamics are described by

$$\dot{x}_i = -a_i(x_i)\left[b_i(x_i) - \sum_j S_j(y_j)m_{ij}\right] \tag{41}$$

$$\dot{y}_j = -a_j(y_j)\left[b_j(y_j) - \sum_i S_i(x_i)m_{ij}\right] \tag{42}$$

$$\dot{m}_{ij} = -m_{ij} + S_i(x_i)\,S_j(y_j) \tag{43}$$

and $a_i > 0$ and $a_j > 0$, and S_i and S_j are bounded monotone increasing ($S_i' > 0$ and $S_j' > 0$) signal functions. At equilibrium, all activation and synaptic velocities are zero.

Proof. Consider the global Lyapunov function L:

$$L = -\sum_i \sum_j S_i S_j m_{ij} + \sum_i \int_0^{x_i} S_i'(\theta_i)\, b_i(\theta_i)\, d\theta_i + \sum_j \int_0^{y_j} S_j'(\epsilon_j)\, b_j(\epsilon_j)\, d\epsilon_j + \tfrac{1}{2}\sum_i \sum_j m_{ij}^2. \tag{44}$$

Then time differentiation and collection of like terms gives

$$\dot{L} = -\sum_i S_i' \dot{x}_i\left[b_i - \sum_j S_j m_{ij}\right] + \sum_j S_j' \dot{y}_j\left[b_j - \sum_i S_i m_{ij}\right] - \sum_i \sum_j \dot{m}_{ij}[S_i S_j - m_{ij}]. \tag{45}$$

Then, using the positivity of a_i and a_j, the terms in braces can be eliminated with the respective equations (41)–(43). This proves that L is strictly decreasing along trajectories:

$$\dot{L} = -\sum_i \frac{S_i'}{a_i}\dot{x}_i^2 - \sum_j \frac{S_j'}{a_j}\dot{y}_j^2 - \sum_i \sum_j \dot{m}_{ij}^2 < 0 \tag{46}$$

for any activation or synaptic change. Since $S_i' > 0$ and $S_j' > 0$, $\dot{L} = 0$ if and only if $\dot{x}_i = \dot{y}_j = \dot{m}_{ij} = 0$ for all i and j. Q.E.D.

The strictly inequality sign in (46) yields asymptotic stability, which ensures that trajectories end in equilibrium points, not merely near them. Asymptotic stability also ensures that the eigenvalues of the Jacobian matrix of the system (41)–(43) have nonpositive real parts near equilibria. A nondegenerate Hessian further ensures that the real parts of the eigenvalues are negative. Then [16] the nonlinear system (41)–(43) converges exponentially quickly as if it were linear.

V. Random Adaptive Bidirectional Associative Memories

Random adaptive bidirectional associative memory (RABAM) models are everywhere perturbed by Brownian diffusions. The differential equations in (41)–(43) now become stochastic differential equations, with random processes as solutions. In the simplest case, Brownian diffusions are simply added to deterministic differential equations. In the more general case adopted here, every activation and synaptic variable represents a separate stochastic process. The stochastic differential equations relate the time evolution of these stochastic processes. Brownian diffusions, or "noise" processes, are then added to the stochastic differential equations. In principle this Ito calculus approach need not preserve the chain rule of deterministic differential calculus. The final section,

though, discusses why for RABAM models the classical chain-rule relationships still hold.

Let B_i, B_j, and B_{ij} be Brownian motion (independent Gaussian increment) processes [35], [41] perturbing the ith neuron in F_X, the jth neuron in F_Y, and the synapse m_{ij}, respectively. The Brownian motions are allowed to have time-varying diffusion parameters. Then the *diffusion* RABAM is described by (47)–(49):

$$dx_i = -a_i(x_i)\left[b_i(x_i) - \sum_j S_j(y_j) m_{ij} \right] dt + dB_i \quad (47)$$

$$dy_j = -a_j(y_j)\left[b_j(y_j) - \sum_i S_i(x_i) m_{ij} \right] dt + dB_j \quad (48)$$

$$dm_{ij} = -m_{ij}\, dt + S_i(x_i)\, S_j(y_j)\, dt + dB_{ij}. \quad (49)$$

The signal Hebb diffusion law (49) can be replaced with the competitive diffusion law

$$dm_{ij} = S_j(y_j)[S_i - m_{ij}]dt + dB_{ij} \quad (50)$$

if S_j is sufficiently steep. Or it can be replaced with differential Hebb or differential competitive diffusion laws if tighter constraints are imposed. For simplicity, we shall formulate the RABAM model in the signal Hebb case only. The extensions to competitive and differential learning proceed exactly as the above extensions of the ABAM theorem. **All RABAM results, like all ABAM results, also immediately extended to high-order systems of arbitrarily high order.**

The RABAM model can be restated in more familiar, less rigorous, "noise notation." Intuitively independent zero-mean noise is added to the ABAM model. The stochastic differential equations then describe the time evolution of network "signals plus noise." This implicitly means that the noise processes are independent of the nonlinear "signal" processes. For emphasis, though, we explictly make the weaker assumption that the noise processes are *uncorrelated* with the "signal" processes. We further assume that the noise processes have finite variances, though they may be time varying. Then the *noise* RABAM model is described by the stochastic differential equations

$$\dot{x}_i = -a_i(x_i)\left[b_i(x_i) - \sum_j S_j(y_j) m_{ij} \right] + n_i \quad (51)$$

$$\dot{y}_j = -a_j(y_j)\left[b_j(y_j) - \sum_i S_i(x_i) m_{ij} \right] + n_j \quad (52)$$

$$\dot{m}_{ij} = -m_{ij} + S_i(x_i)\, S_j(y_j) + n_{ij} \quad (53)$$

$$E(n_i) = E(n_j) = E(n_{ij}) = 0 \quad (54)$$

$$V(n_i) = \sigma_{n_i}^2 < \infty, \quad \sigma_j^2 < \infty, \quad \sigma_{ij}^2 < \infty. \quad (55)$$

Noise can be added within the general b_i and b_j terms, perhaps reflecting random input signals. A separate analysis [34] shows that additive input noise can be accommodated for additive and shunting activation models. For additive activation models, such additive activation noise can be included in the noise terms n_i and n_j.

Will so much noise destabilize the system? So much noise with so much feedback would seem to promote chaos, especially since the network dimensions n and p can be arbitrarily large. How can stable learning occur?

The RABAM theorem ensures stochastic stability. Nonlinear interactions suppress noise and suppress it exponentially quickly. In effect, RABAM equilibria are ABAM equilibria that randomly vibrate. The diffusion parameters, or the noise variances, control the range of vibration. Average RABAM behavior is just ABAM behavior. Since noise perturbations do not destroy equilibria, the RABAM theorem says that unsupervised learning is structurally stable in the stochastic sense. The result applies with equal force, though with less theoretical interest, for unsupervised learning in feedforward networks.

The RABAM theorem can be motivated with a simple thought experiment or, better, a few hand calculations. Consider a discrete additive BAM with fixed matrix M. Find its bipolar fixed points in the product space $\{-1, 1\}^n \times \{-1, 1\}^p$. Now add a small amount of zero-mean noise to each memory element m_{ij}. Since a discrete BAM signal function is a threshold function, it is unlikely that more than very few neurons, if any, change state differently during iterations than they did before. It is even less likely that they will do so as n and p increase. The same fixed points tend to be reached, and tend to persist once reached. This corresponds to adding noise at the synaptic level. Now repeat the computation, but also add zero-mean noise to each neuron's activation at each iteration. Then repeat this computation, adding new noise to the matrix M each time. This allows the synaptic noise processes to be "lower" than the neuronal noise processes. Again the threshold signal functions make it unlikely that the signal patterns will change significantly, if at all, during iterations or in equilibrium.

A. RABAM Theorem

The RABAM model (47)–(50), or (51)–(55) is globally stable. If signal functions are strictly increasing and amplification functions a_i and a_j are strictly positive, the RABAM model is asymptotically stable.

Proof. The ABAM Lyapunov function (44) is now a random process. At each time t, $L(t)$ is a random variable. We conjecture that *the expected ABAM Lyapunov function $E(L)$ is a Lyapunov function for the RABAM system*, where the expectation is with respect to all random parameters:

$$E(L) = \int \cdots \int L\, p(X, Y, M)\, dX\, dY\, dM. \quad (56)$$

(Recall that each activation and synaptic parameter represents a random process separate from the random process got simply by adding noise to a deterministic variable.)

The proof strategy is to replace the time derivative of the expectation with the expectation of the time derivative

of the ABAM Lyapunov function, which we calculated above. Technically we need to assume sufficient smoothness conditions on the RABAM model to bring the time derivative inside the multiple integrals in (56). This assumption adds little burden. Then

$$\begin{aligned}
\dot{E}(L) &= E(\dot{L}) \text{ and by (45)} \\
&= E\Bigg\{ \sum_i S_i' \dot{x}_i \Big[b_i - \sum_j S_j m_{ij} \Big] \\
&\quad + \sum_j S_j' \dot{y}_j \Big[b_j - \sum_i S_i m_{ij} \Big] \\
&\quad - \sum_i \sum_j \dot{m}_{ij} [-m_{ij} + S_i S_j] \Bigg\} \\
&= E\Bigg\{ - \sum_i S_i' a_i \Big[b_i - \sum_j S_j m_{ij} \Big]^2 \\
&\quad - \sum_j S_j' a_j \Big[b_j - \sum_i S_i m_{ij} \Big]^2 \\
&\quad - \sum_i \sum_j [-m_{ij} + S_i S_j]^2 \Bigg\} \\
&\quad + \sum_i E\Big\{ n_i S_i' \Big[b_i - \sum_j S_j m_{ij} \Big] \Big\} \\
&\quad + \sum_j E\Big\{ n_j S_j' \Big[b_j - \sum_i S_i m_{ij} \Big] \Big\} \\
&\quad - \sum_i \sum_j E\{ n_{ij} [-m_{ij} + S_i S_j] \} \qquad (57)
\end{aligned}$$

upon eliminating the activation and synaptic velocities in (57) with the RABAM dynamical equations (51)–(53)

$$\begin{aligned}
&= E[\dot{L}_{\text{ABAM}}] + \sum_i E(n_i) E\Big\{ S_i' \Big[b_i - \sum_j S_j m_{ij} \Big] \Big\} \\
&\quad + \sum_j E(n_j) E\Big\{ S_j' \Big[b_j - \sum_i S_i m_{ij} \Big] \Big\} \\
&\quad - \sum_i \sum_j E(n_{ij}) E[-m_{ij} + S_i S_j] \qquad (58)
\end{aligned}$$

by the uncorrelatedness (independence) of the "signal" and additive noise terms in the RABAM model, and by the facts that S_i' and S_j' are nonnegative functions of x_i and y_j respectively, and a_i and a_j are nonnegative essentially arbitrary functions (so $S_i' = a_i$ and $S_j' = a_j$ possible)

$$= E[\dot{L}_{\text{ABAM}}]$$

by (54). So $\dot{E}(L) \leq 0$ or $\dot{E}(L) < 0$ along trajectories according as $\dot{L}_{\text{ABAM}} \leq 0$ or $\dot{L}_{\text{ABAM}} < 0$. Q.E.D.

VI. Noise-Saturation Dilemma and the RABAM Noise Suppression Theorem

How much do RABAM trajectories and equilibria vibrate? To answer this question we need to examine the *second-order* behavior of the RABAM model. This behavior depends fundamentally on the variances of the additive noise processes. Observe that the zero-mean assumption (54) implies that the time-varying "variances" σ_i^2, σ_j^2, and σ_{ij}^2 are the respective instantaneous mean-squared "noises" $E(n_i^2)$, $E(n_j^2)$, and $E(n_{ij}^2)$, since in general $V(x) = E(x^2) - E^2(x)$.

Observed RABAM second-order behavior consists of the observed instantaneous *mean-squared velocities* $E(\dot{x}_i^2)$, $E(\dot{y}_j^2)$, and $E(\dot{m}_{ij}^2)$. The mean-squared velocities measure the magnitude of instantaneous RABAM change. They are at least as large as the underlying instantaneous "variances" of the activation velocity and synaptic velocity processes, since, for example

$$E(\dot{x}_i^2) \geq E(\dot{x}_i^2) - E^2(\dot{x}_i) = V(\dot{x}_i). \qquad (59)$$

Intuitively the mean-squared velocities should depend on the instantaneous "variances" of the noise processes in (51)–(53). The more the noise processes hop about their means, the greater the potential for the activations and synapses to change state. But this intuition seems to run counter to the structural stability established by the RABAM theorem. Surely, it seems, if the magnitudes of the noise fluctuations grow arbitrarily large, there comes a point—and perhaps a point quickly reached in the midst of massive noisy feedback—where the RABAM system transitions from stability to instability.

The RABAM noise suppression theorem guarantees that no noise processes can destabilize a RABAM if the noise processes have *finite* instantaneous variances. (Cauchy noise, for example, in theory could destabilize a RABAM since it has infinite variance. In practice, though, even Cauchy variance is finite, and so it will never destabilize a RABAM.) Preliminary simulations [43], where noise fluctuations are many orders of magnitude greater than activation and synaptic fluctuations, have confirmed this surprising prediction. In some sense noise cannot beat RABAM stability. Moreover, the RABAM noise suppression theorem ensures that noise will be "quenched," to use Grossberg's term [13], exponentially quickly in most cases.

To prove the RABAM noise suppression theorem, we must make explicit how RABAM instantaneous mean-squared velocities depend on the underlying instantaneous noise variances. The following lemma grounds the intuition that observed second-order behavior—the instantaneous mean-squared velocities—involves at least as much fluctuation as is found in the noise itself.

Lemma:

$$E(\dot{x}_i^2) \geq \sigma_i^2, \quad E(\dot{y}_j^2) \geq \sigma_j^2, \quad E(\dot{m}_{ij}^2) \geq \sigma_{ij}^2. \qquad (60)$$

Proof. All three inequalities are proved by squaring both sides of the RABAM equations (51)–(53), taking expectations, and using (54) and the fact that the noise is uncorrelated with the additive nonlinear "signal" terms. Q.E.D.

It is not true that the squared velocity processes are never less than the squared noise processes at every instant. It is only true on average at every instant.

Grossberg's *noise-saturation dilemma* [13] motivates the use of the term "noise suppression" in the RABAM corollary below. The noise-saturation dilemma asks how neurons can have an effective infinite dynamical range when they operate between upper and lower bounds and yet not treat small input signals as noise: "If the x_i are sensitive to large inputs, then why do not small inputs get lost in internal system noise? If the x_i are sensitive to small inputs, then why do they not all saturate at their maximum values in response to large inputs?" [14] This vexing and ubiquitous dilemma, it even confronts the salesperson who trys to balance her presentation between "little" and "big" customers, is the supreme motivator behind Grossberg's shunting-model perspective of neural networks.

Grossberg resolves the saturation half of the dilemma by showing [13], as mentioned above, that shunting models remain sensitive to relative pattern information over a wide range of inputs. He also shows that additive models quickly saturate to upper bounds for large inputs. Indeed this saturation invariance result is arguably Grossberg's greatest achievement. Besides giving information-processing insights into the global dynamics of Hodgkin–Huxley type networks, it also drives Grossberg's conception and implementation of ART behavior, and is at the heart of his recent vision theory. On the other hand, as Carver Mead and other neural VLSI designers have observed, it is well known that a simple logarithmic transduction of local input light intensity into electric potential in the visual system achieves in one stroke both sensitivity to input light intensities over many orders of magnitude and "discounts the illuminant" [14] by equating voltage differences to logarithms of intensity ratios.

Grossberg's resolution of the noise half of the noise-saturation dilemma is far less satisfactory. Grossberg [13] argues that noisy patterns are uniform input patterns and that, for a particular small threshold value, uniform noise is "suppressed" by all neurons in the field shutting off. Besides the dependence on a specific noise threshold, this argument is objectionable on at least two counts. First, noise permeates all parameters and all signals and certainly need not be uniform. Grossberg admits this in his above description of the noise-saturation dilemma when he asks why small inputs do not "get lost in internal system noise." System noise makes everything "jiggle," including relative input pattern values. This is the noise modeled by the additive noise processes in the RABAM equations (51)-(53) or, more realistically, by the additive diffusion processes in the diffusion RABAM equations (47)-(49).

Second, shutting off neurons to suppress noise seems akin to curing the patient by killing him. The goal is to continue "computing" as accurately as possible no matter how noisy the environment. Background noise can be high in feedback systems where noise can multiply by recirculating. In fairness, Grossberg [14] argues that special classes of signal functions, especially sigmoid signal functions, help quench pattern noise by contrast-enhancing input signals. Signal function nonlinearities surely help suppress this special occurrence of noise. But what about synaptic noise? What about joint synaptic and activation noise? What about noise compounded by feedback? How do we know such pervasive noise will not prevent an ART system from adaptively resonating, or ruin an adaptive-resonance equilibrium once achieved?

The RABAM noise suppression theorem is an alternative resolution of the noise half of the noise-saturation dilemma. It guarantees that second-order behavior in RABAM systems is as good as it can be: mean-square velocities decrease exponentially quickly to their lower bounds. As the above lemma shows, these lower bounds are just the underlying driving noise variances. Thus the observed fluctuations, the mean-squared velocities, track the unobserved noise fluctuations. Unaided feedback intuitions might easily lead to the prediction that, in light of the lemma, mean-squared velocities may tend toward infinity, especially for widely fluctuating noise processes.

A. RABAM Noise Suppression Theorem

For strictly increasing signal functions S_i and S_j, positive amplification functions a_i, and nondegenerate Hessian conditions: as the RABAM system (51)-(55) converges exponentially quickly, mean-squared activation and synaptic parameters decrease to their lower bounds exponentially quickly:

$$E(\dot{x}_i^2) \downarrow \sigma_i^2, \quad E(\dot{y}_j^2) \downarrow \sigma_j^2, \quad E(\dot{m}_{ij}^2) \downarrow \sigma_{ij}^2. \tag{61}$$

Proof. The proof uses the asymptotic convergence established in the above RABAM theorem for the monotonicity and positivity assumptions and the lower bound on mean-square velocities established in the lemma (60). Then

$$\dot{E}(L) = E(\dot{L})$$

$$= E\left[\sum_i \frac{S_i' \dot{x}_i (n_i - \dot{x}_i)}{a_i} + \sum_j \frac{S_j' \dot{y}_j (n_j - \dot{y}_j)}{a_j} - \sum_i \sum_j \dot{m}_{ij} (n_{ij} - \dot{m}_{ij}) \right]$$

by using the positivity of the amplification functions and (51)-(53) to eliminate the terms in braces in (57) in the proof of the RABAM theorem

$$= E\left[\sum_i \frac{S_i'}{a_i} \dot{x}_i^2 - \sum_j \frac{S_j'}{a_j} \dot{y}_j^2 - \sum_i \sum_j \dot{m}_{ij}^2 \right] + E\left[\sum_i \frac{S_i'}{a_i} \dot{x}_i n_i + \sum_j \frac{S_j'}{a_j} \dot{y}_j n_j + \sum_i \sum_j \dot{m}_{ij} n_{ij} \right]$$

$$= E[\dot{L}_{\text{ABAM}}] - \sum_i E(n_i) E\left[S_i' \left(b_i - \sum_j S_j m_{ij} \right) \right] - \sum_j E(n_j) E\left[S_j' \left(b_j - \sum_i S_i m_{ij} \right) \right] - \sum_i \sum_j E(n_{ij}) E[-m_{ij} + S_i S_j] + E\left[\sum_i \frac{S_i'}{a_i} n_i^2 + \sum_j \frac{S_j'}{a_j} n_j^2 + \sum_i \sum_j n_{ij}^2 \right]$$

by using (51)–(53) again to eliminate activation and synaptic velocities in the second expectation above, rearranging, and, as in the proof of the RABAM theorem, using the uncorrelatedness of noise and "signal" terms in (51)–(53) as discussed above to obtain (59)

$$= -\sum_i E\left[\frac{S_i'}{a_i}(\dot{x}_i^2 - n_i^2)\right] - \sum_j E\left[\frac{S_j'}{a_j}(\dot{y}_j^2 - n_j^2)\right] - \sum_i \sum_j (E(\dot{m}_{ij}^2) - \sigma_{ij}^2) \quad (62)$$

by the zero-mean noise assumption (54) and rearrangement. The lemma ensures that the double sum is nonnegative. The RABAM theorem establishes that the Lyapunov function $E(L)$ strictly decreases along trajectories, and thus trajectories end at equilibrium points and arrive there exponentially quickly. This, together with the positivity (and well behavedness [34]) of the weight ratios S'/a, yields the equilibrium conditions:

$$E(\dot{x}_i^2) = \sigma_i^2, \quad E(\dot{y}_j^2) = \sigma_j^2, \quad E(\dot{m}_{ij}^2) = \sigma_{ij}^2. \quad (63)$$

This implies (62). Q.E.D.

The RABAM noise suppression generalizes the equilibrium conditions obtained in the ABAM theorem in the asymptotic-convergence case. For if the instantaneous "variances" in (63) are zero, then [38] the squared velocities, and thus the velocities, are zero almost everywhere. The zero-variance case is the deterministic case. The sigma-algebra of the probability space is degenerate; it only contains the whole space and the null set. Thus the activation and synaptic velocities are zero everywhere, as in the strict ABAM case. Also note that throughout the proofs of the RABAM theorem and the RABAM noise suppression theorem, the synaptic terms are easier to work with, and the results are "cleaner," because they do not possess nonlinear signal and amplification terms. We recall again that the above two theorems are also valid for suitably randomized competitive, differential Hebb, and differential competitive learning laws under appropriate conditions.

VII. RABAM Annealing and the Ito-Stratonovich Stochastic Calculus

Gradient systems are globally stable. The above theorems are an extension of this general Lyapunov fact. For example, Cohen and Grossberg [6] showed that their symmetric nonlearning autoassociative system can be written in pseudogradient form for monotone increasing signal functions and positive amplification functions.

Geman and Hwang [8] recently showed that stochastic gradient systems with scaled additive Brownian diffusions (noise) perform *simulated annealing* in a weak sense. The gradient is formed from a cost function to be searched by scaled random hill climbing. If the noise is initially scaled high enough (to a physically unrealizable size), then gradually decreasing the nonnegative "temperature" $T(t)$ scaling factor can bounce the system state out of local minima and trap it in global minima. The convergence, though, must proceed exponentially slowly and is only gous to the convergence in distribution found in central limit theorems). The result is not true for convergence with probability one or even convergence probability. There is still some probability that the system state will bounce out of global or near-global minima as "cooling" finishes.

We now extend the RABAM theorem and RABAM noise suppression theorem to include simulated annealing in the general Geman–Hwang sense. For this we introduce the activation "temperatures" or annealing schedules $T_i(t)$ and $T_j(t)$ and the synaptic schedules $T_{ij}(t)$. The temperatures are nonnegative deterministic functions. So they can be brought outside all expectations in proofs. The RABAM annealing model is more general than the Geman–Hwang gradient model, and vastly more general than popular additive-activation annealing models, because learning is permitted and because learning too can be annealed, although perhaps at a different rate than activation annealing. The RABAM annealing model is defined by scaling the diffusion differentials in (47)–(49) with the square root of the corresponding annealing schedules or, in the noise RABAM, by replacing (51)–(53) with (64)–(66):

$$\dot{x}_i = -a_i\left[b_i - \sum_j S_j m_{ij}\right] + \sqrt{T_i}n_i \quad (64)$$

$$\dot{y}_j = -a_j\left[b_j - \sum_i S_i m_{ij}\right] + \sqrt{T_j}n_j \quad (65)$$

$$\dot{m}_{ij} = -m_{ij} + S_i S_j + \sqrt{T_{ij}}n_{ij} \quad (66)$$

where again (67) can be replaced with the other unsupervised learning laws discussed above with appropriate additional constraints.

A. RABAM Annealing Theorem

The RABAM annealing model is globally stable, and asymptotically stable for monotone increasing signal functions and positive amplification functions, in which case the mean-squared activation and synaptic velocities decrease to their temperature-scaled instantaneous "variances" exponentially fast:

$$E(\dot{x}_i^2) \downarrow T_i\sigma_i^2, \quad E(\dot{y}_j^2) \downarrow T_j\sigma_j^2, \quad E(\dot{m}_{ij}^2) \downarrow T_{ij}\sigma_{ij}^2. \quad (67)$$

Proof. The proof largely duplicates the proofs of the RABAM theorem and RABAM noise suppression theorem. Again $E(L)$ is a sufficiently smooth Lyapunov function that allows time differentiation of the integrand. When the diffusion or noise RABAM annealing equations are used to eliminate activation and synaptic velocities in the time-differentiated Lyapunov function, the resulting temperature functions that occur can be factored outside all expectations. The nonnegativity of the temperature functions keeps them from affecting the structure of expanded time derivative of $E(L)$. The random weight functions S'/a are assumed sufficiently well behaved to keep the expectations in which they occur nonnegative. The above

the mean-squared velocity $E(\dot{x}_i^2)$ is bounded below by $T_i\sigma_i^2$. Then, (62) is generalized to

$$\dot{E}(L) = -\sum_i E\left[\frac{S_i'}{a_i}(\dot{x}_i^2 - T_i n_i^2)\right] - \sum_j E\left[\frac{S_j'}{a_j}(\dot{y}_j^2 - T_j n_j^2)\right] - \sum_i \sum_j (E(\dot{m}_{ij}^2) - T_{ij}\sigma_{ij}^2). \quad (68)$$

Q.E.D.

The RABAM annealing theorem is a nonlinear and continuous generalization of Boltzmann machine learning [40], provided learning is Hebbian and very slow. The Boltzmann machine uses discrete symmetric additive autoassociative dynamics. Binary neurons are annealed during periods of Hebbian and anti-Hebbian learning. Here Hebbian learning corresponds to (66) with $T_{ij}(t) = 0$ for all t. Anti-Hebbian learning further replaces the Hebb product S_iS_j in (66) with the negative product $-S_iS_j$. Anti-Hebbian learning (during "free-running" training [40]) can in principle destabilize a RABAM system. This is less likely to occur, though, the slower the anti-Hebbian learning. (The activation terms in the time derivative of $E(L)$ stay negative and can outweigh the possibly positive anti-Hebbian terms, even if learning is fast.) Incidental instability perhaps is not even a problem in this phase of annealing, since the intention is to undo some of the learning in the "environmental" annealing phase. The fundamental distinction between unsupervised RABAM learning and temperature-supervised annealing learning is how noise is treated. Simulated annealing systems search or learn with noise. Unsupervised RABAM systems learn despite noise. During "cooling," the continuous annealing schedules define the flow of RABAM equilibria in the product state space of continuous nonlinear random processes. Equation (67) implies that no finite temperature value, however large, can destabilize a RABAM.

Finally, the proofs of the above RABAM theorems repeatedly use the familiar chain rule of differential calculus. In general, the chain rule does not apply to systems of nonlinear stochastic differential equations, at least not in the general case where each nonlinear parameter is itself a stochastic process. This is the general setting for the Ito calculus. One exception is the related Stratonovich calculus, which defines a stochastic integral (an integral defined with respect to a random measure [41] with as lightly different partitioning of the time interval. The Stratonovich calculus includes the classical chain rule, though in general at the expense of possessing non-Markovian solution processes.

Maybeck [35] shows that, with probability one, the Ito stochastic differential equals the Stratonovich stochastic differential plus a term involving the nonlinear random scaling factor on the underlying Brownian diffusion. The two differentials and corresponding integrals are equal when this extra term is zero. This is fortunately always true for RABAM systems since noise terms are scaled with constants or sequences of constants (deterministic annealing schedules). The extra term involves the derivative of this constant with respect to the corresponding random activation or synapse. Thus RABAM models enjoy the best of both stochastic-calculus worlds. They maintain the familiar chain rule of Stratonovich stochastic dynamical systems and inherit the better-explored properties of Ito stochastic dynamical systems. For instance, all RABAM solution processes are Markov processes. This promises a new approach to nonlinear stochastic optimal estimation and control.

VIII. Conclusions

The RABAM model unifies many popular feedforward and feedback unsupervised learning systems and extends them to the more realistic, and more complex, random process domain. Unsupervised learning is structurally stable for wide families of nonlinear feedback dynamical systems. This holds for the popular signal Hebb and competitive learning feedback systems under quite general conditions. It holds to a lesser extent for the largely unexplored signal-velocity learning feedback systems that adapt with differential Hebb or differential competitive laws. Pulse-coded [10], [11] signal functions augment the class of feedback systems that can stably learn with the differential Hebb and differential competitive laws, since in this case they give back, respectively, signal Hebb and competitive learning behavior much of the time. The pulse-coding framework also promises new engineering approaches to implementing adaptive networks, perhaps with sinusoidal techniques, as well as suggesting new roles for signal-velocity synaptic mechanisms in real neural systems. The feedback in these stable dynamical systems can always be eliminated to produce unsupervised feedforward systems that stably learn with Hebbian, competitive, or signal-velocity learning laws.

The stability of RABAM models yields the structural stability of ABAM models. From an engineering perspective, this means we can more confidently build large-scale ABAM networks with electrical, optical, electrooptic, and perhaps other (molecular, fluid, plasma, polymer, etc.) devices.

For the neurobiologist, the structural stability of ABAM models suggests that at least some of the consistent criticism that neural models are "unrealistic" is unfounded. The many intricate neuronal and molecular properties that the neurobiologist studies, and finds missing in neural network models, are modeled in RABAM systems as random unmodeled effects. The RABAM noise suppression theorem says these unmodeled effects are ignored by the network's global computations almost as quickly as they are encountered. Like many quantum-level effects in electrical devices, these unmodeled effects simply do not affect the structure of global network computations—so long as they are net random effects.

How plausible is this? Some unmodeled effects of course depend on neuronal and synaptic behavior and so are not accurately modeled as independent noise processes, though perhaps central-limit (Gaussian) effects emerge from the interaction of many such processes. Many correlated effects can also be incorporated as slowly

varying parameters in the "signal" part of the RABAM model.

In general, the sheer number (sample size) of unmodeled effects suggests a Brownian approximation. To the extent that the unmodeled synaptic and neuronal effects involve many independently interacting continuous phenomena, the net result is a Brownian diffusion, as assumed by RABAM models. This is because finite-variance continuous processes with independent increments in time have Gaussian increments [35], and hence give rise to a Brownian diffusion.

ACKNOWLEDGMENT

The author thanks F. Watkins for many valuable technical discussions and conjecture-testing simulations.

REFERENCES

[1] J. A. Anderson, J. W. Silverstein, S. R. Ritz, and R. S. Jones, "Distinctive features, categorical perception, and probability learning: Some applications of a neural model," *Psycholog. Rev.*, vol. 84, pp. 413-451, 1977.

[2] J. A. Anderson, "Cognitive and psychological computation with neural models," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, pp. 799-815, Sept./Oct. 1983.

[3] G. A. Carpenter, M. A. Cohen, and S. Grossberg, "Computing with neural networks," *Science*, vol. 235, pp. 1226-1227, Mar. 6, 1987.

[4] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Comput. Vision, Graph., Image Process.*, vol. 37, pp. 54-115, 1987.

[5] G. A. Carpenter and S. Grossberg, "ART 2: Self-organization of stable category recognition codes for analog input patterns," *Appl. Opt.*, vol. 26, no. 23, pp. 4919-4930, Dec. 1, 1987.

[6] M. A. Cohen and S. Grossberg, "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, pp. 815-826, Sept./Oct. 1983.

[7] M. A. Cohen and S. Grossberg, "Masking fields: A massively parallel neural architecture for learning, recognizing, and predicting multiple groupings of patterned data," *Appl. Opt.*, vol. 26, p. 1866, 1987.

[8] S. Geman and C. Hwang, "Diffusions for global optimization," *SIAM J. Contr. Optimiz.*, vol. 24, no. 5, pp. 1031-1043, Sept. 1986.

[9] R. Gilmore, *Catastrophe Theory for Scientists and Engineers*. New York: Wiley, 1981.

[10] M. A. Gluck, D. B. Parker, and E. Reifsnider, "Some biological implications of a differential-Hebbian learning rule," *Psychobiology*, vol. 16, no. 3, pp. 298-302, 1988.

[11] M. A. Gluck, D. B. Parker, and E. S. Reifsnider, "Learning with temporal derivatives in pulse-coded neuronal systems," in *Proc. Nov. 1988 IEEE Neural Inform. Proc. Syst. (NIPS) Conf.* (Denver, CO) San Mateo, CA: Morgan Kaufman, in press.

[12] S. Grossberg, "On learning and energy entropy dependence in recurrent and nonrecurrent signed networks," *J. Statistic. Phys.*, vol. 1, pp. 319-350, 1969.

[13] S. Grossberg, *Studies of Mind and Brain*. Boston, MA: Reidel, 1982.

[14] S. Grossberg, "Nonlinear neural networks: Principles, mechanisms, and architectures," *Neural Networks*, vol. 1, no. 1, pp. 17-61, 1988.

[15] R. Hecht-Nielsen, "Counterpropagation networks," *Appl. Opt.*, vol. 26, no. 3, pp. 4979-4984, Dec. 1, 1987.

[16] M. W. Hirsch and S. Smale, *Differential Equations, Dynamical Systems, and Linear Algebra*. New York: Academic, 1974.

[17] M. W. Hirsch, *Differential Topology*. New York: Springer-Verlag, 1976.

[18] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *J. Physiology*, vol. 117, pp. 500-544, 1952.

[19] J. J. Hopfield, "Neural networks with graded response have collective computational properties like those of two-state neurons," *Proc. Nat. Acad. Sci., U.S.*, vol. 81, pp. 3088-3092, 1984.

[20] J. M. Kinser, H. J. Caulfield, and J. Shamir, "Design for a massive all-optical bidirectional associative memory: The big BAM," *Appl. Opt.*, vol. 27, no. 16, pp. 3442-3443, Aug. 15, 1988.

[21] A. H. Klopf, "A drive-reinforcement model of single neuron function: An alternative to the Hebbian neuronal model," *Proc. Amer. Inst. Phys.: Neural Networks for Computing*, pp. 265-270, Apr. 1986.

[22] A. H. Klopf, "Drive-reinforcement learning: A real-time learning mechanism for unsupervised learning," in *Proc. IEEE Int. Conf. Neural Networks (ICNN-87)*, vol. II, June 1987, pp. 441-445.

[23] A. H. Klopf, "A neuronal model of classical conditioning," *Psychobiology*, vol. 16, no. 2, pp. 85-125, 1988.

[24] T. Kohonen, *Self-Organization and Associative Memory*, second ed. New York: Springer-Verlag, 1988.

[25] B. Kosko, "Adaptive inference," monograph (Verac, Inc., tech rep.), June 1985.

[26] B. Kosko and J. S. Limm, "Vision as causal activation and association," *Proc. SPIE Int. Soc. Opt. Eng.*, vol. 579, pp. 104-109, Sept. 1985.

[27] B. Kosko, "Differential Hebbian learning," *Proc. Amer. Inst. Phys. Neural Networks for Computing*, pp. 277-282, Apr. 1986.

[28] B. Kosko and C. C. Guest, "Optical bidirectional associative memories," *Proc. SPIE Int. Soc. Opt. Eng.*, vol. 758, p. 11, 1987.

[29] B. Kosko, "Competitive adaptive bidirectional associative memories," in *Proc. IEEE Int. Conf. Neural Networks (ICNN-87)*, vol. II, June 1987, pp. 261-268.

[30] B. Kosko, "Adaptive bidirectional associative memories," *Appl. Opt.*, vol. 26, no. 23, pp. 4947-4960, Dec. 1, 1987.

[31] B. Kosko, "Bidirectional associative memories," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-18, pp. 49-60, Jan./Feb. 1988.

[32] B. Kosko, "Feedback stability and unsupervised learning," *Proc. 2nd IEEE Int. Conf. Neural Networks (ICNN-88)*, vol. I, July 1988, pp. 141-152.

[33] B. Kosko, "Hidden patterns in combined and adaptive knowledge networks," *Int. J. Approx. Reason.*, vol. 2, no. 4, pp. 377-393, Oct. 1988.

[34] B. Kosko, "Structural stability of unsupervised learning in feedback networks," submitted for publication.

[35] P. S. Maybeck, *Stochastic Models, Estimation, and Control*, vol. 2. New York: Academic, 1982.

[36] T. S. Parker and L. O. Chua, "Chaos: A tutorial for engineers," *Proc. IEEE*, vol. 75, no. 8, pp. 982-1008, Aug. 1987.

[37] W. K. Pratt, *Digital Image Processing*. New York: Wiley, 1978.

[38] W. Rudin, *Real and Complex Analysis*, second ed. New York: McGraw-Hill, 1974.

[39] D. E. Rumelhart and D. Zipser, "Feature discovery by competitive learning," *Cognitive Science*, vol. 9, pp. 75-112, 1985.

[40] D. E. Rumelhart, G. E. Hinton, and J. L. McClelland, "A general framework for parallel distributed processing," in *Parallel Distributed Processing*, vol. 1, D. E. Rumelhart, J. L. McClelland, Eds. Cambridge, MA: M.I.T. Press, 1986, ch. 2.

[41] A. V. Skorokhod, *Studies in the Theory of Random Processes*. Reading, MA: Addison-Wesley, 1965.

[42] R. Thom, *Structural Stability and Morphogenesis*. Reading, MA: Addison-Wesley, 1975.

[43] F. Watkins, personal communication.

Bart Kosko (M'85) received the bachelor's degrees in philosophy and economics from the University of Southern California in 1982, the master's degree in applied mathematics from the University of California, San Diego, in 1983, and the Ph.D. degree in electrical engineering from the University of California, Irvine, in 1987.

He is an Assistant Professor of Electrical Engineering at the University of Southern California. He is also Managing Editor of the Springer-Verlag monograph series *Lecture Notes in Neural Computing* and is Associate Editor of *Neural Networks*, IEEE TRANSACTIONS ON NEURAL NETWORKS, *Journal of Mathematical Biology*, and *Lecture Notes in Biomathematics*.

Dr. Kosko is an elected governor of the Internal Neural Network Society and is a USC Shell Oil Faculty Fellow. He is Program Co-chairman of the Summer 1990 International Joint Conference on Neural Networks. He was Program Chairman of the 1988 IEEE International Conference on Neural Networks (ICNN-88) and Program and Organizing Chairman of the first IEEE ICNN-87.

Int. J. General Systems, Vol. 17, pp. 211–240
Reprints available directly from the publisher
Photocopying permitted by license only

FUZZINESS VS. PROBABILITY

BART KOSKO

Electrical Engineering Department, Signal and Image Processing Institute, University of Southern California, Los Angeles, California 90089-0272, USA

(*Received 14 November 1989; in final form 2 March 1990*)

So far as the laws of mathematics refer to reality, they are not certain. And so far as they are certain, they do not refer to reality.

Albert Einstein

Fuzziness is explored as an alternative to randomness for describing uncertainty. The new *sets-as-points* geometric view of fuzzy sets is developed. This view identifies a fuzzy set with a point in a unit hypercube and a nonfuzzy set with a vertex of the cube. Paradoxes of two-valued logic and set theory, such as Russell's paradox, correspond to the midpoint of the fuzzy cube. The fundamental questions of fuzzy theory—How fuzzy is a fuzzy set? How much is one fuzzy set a subset of another?—are answered geometrically with the Fuzzy Entropy Theorem, the Fuzzy Subsethood Theorem, and the Entropy-Subsethood Theorem. A new geometric proof of the Subsethood Theorem is given, a corollary of which is that the apparently probabilistic relative frequency n_A/N turns out to be the deterministic subsethood $S(X, A)$, the degree to which the sample space X is contained in its subset A. So the frequency of successful trials is viewed as the degree to which all trials are successful. Recent Bayesian polemics against fuzzy theory are examined in light of the new sets-as-points theorems.

INDEX TERMS: Probability Theory, fuzzy set theory, fuzzy subsethood, geometry of fuzzy sets.

1. FUZZINESS IN A PROBABILISTIC WORLD

Is uncertainty the same as randomness? If we are not sure about something, is it only up to chance? Do the notions of likelihood and probability exhaust our notions of uncertainty?

Many people, trained in probability and statistics, believe so. Some even say so, and say so loudly. These voices are often heard in the Bayesian camp of statistics, where probability is viewed, not as a frequency or other objective testable quantity, but as a subjective *state of knowledge*.

Bayesian physicist E. T. Jaynes says[6] that

> any method of inference in which we represent degrees of plausibility by real numbers, is necessarily either equivalent to Laplace's [probability], or inconsistent.

He claims physicist R. T. Cox[3] has proven this as a theorem, a claim we examine below.

More recently, Bayesian statistician Dennis Lindley[13] issued an explicit challenge:

probability is the only sensible description of uncertainty and is adequate for all problems involving uncertainty. All other methods are inadequate.

Lindley directs his challenge in large part at *fuzzy theory*, the theory that *all things admit degrees*, but admit them deterministically. This article accepts the probabilist's challenge from the fuzzy viewpoint—admitting but ignoring other approaches to uncertainty, such as Dempster–Shafer belief function theory—by defending fuzziness from new geometric first principles and by questioning the reasonableness and the axiomatic status of randomness. The new view is the *sets-as-points view*[11] of fuzzy sets: A fuzzy set is a point in a unit hypercube and a nonfuzzy set is a corner of the hypercube.

There are conceptual and theoretical differences between randomness and fuzziness. Some can be illustrated with examples. Some can be proven with theorems, as we show below.

There are also many similarities. The chief, but superficial, similarity is that both systems describe uncertainty with numbers in the unit interval $[0, 1]$. This ultimately means that both systems describe uncertainty numerically. The structural similarity is that both systems combine sets and propositions associatively, commutatively, and distributively. The key distinction concerns how the systems deal simultaneously with a thing A and its opposite A^c.

Questions raise doubt, and doubt suggests room for change. So to commence the exposition, consider the following two questions, one fuzzy and the other probabilistic:

i) Is it always and everywhere true that $A \cap A^c = \emptyset$?

ii) Who can *derive* the conditional probability operator

$$P(B|A) = \frac{P(A \cap B)}{P(A)}?$$

The second question may appear less fundamental than the first question, which asks whether fuzziness exists. The Entropy-Subsethood Theorem below shows that the first and second questions are connected: How fuzzy a fuzzy set A is can be measured by how much the superset $A \cup A^c$ is a subset of its own subset $A \cap A^c$, a paradoxical relationship unique to fuzzy theory. In contrast, in probability theory this state of affairs is impossible (has zero probability): $P(A \cap A^c | A \cup A^c) = P(\emptyset | X) = 0$, where X is the sample space or "sure event" and the empty set $\emptyset$ is the "impossible event".

The conditioning or subsethood in the second question is at the heart of Bayesian probabilistic systems. The absence of a first-principles derivation of $P(B|A)$ in itself may be acceptable. One simply agrees to take the ratio relationship as an axiom. The problem is that the new sets-as-points view of fuzzy sets *derives* its conditioning operator as a theorem from first principles. The history of science suggests that systems that hold theorems as axioms continue to evolve.

The first question asks whether the law of noncontradiction—one of Aristotle's three "laws of thought" along with the laws of excluded middle, $A \cup A^c = X$, and identity, $A = A$—can be violated. Set fuzziness occurs when, and only when, it is violated. Classical logic and set theory assume that the law of noncontradiction,

and equivalently the law of excluded middle, is never violated. That is what makes the classical theory black or white. Fuzziness begins where Western logic ends.

2. RANDOMNESS VS. AMBIGUITY: WHETHER VS. HOW MUCH

Fuzziness describes *event ambiguity*. It measures the degree to which an event occurs, not whether it occurs. Randomness describes the uncertainty of *event occurrence*. An event occurs or not, and you can bet on it. At issue is the nature of the occurring event: whether it itself is uncertain in any way, in particular whether it can be unambiguously distinguished from its opposite.

Whether an event occurs is "random". To what degree it occurs is fuzzy. Whether an ambiguous event occurs—as when we say there is 20% chance of light rain tomorrow—involves compound uncertainties, the probability of a fuzzy event.

In practice we regularly apply probabilities to fuzzy events: small errors, satisfied customers, A students, safe investments, developing countries, noisy signals, spiking neurons, dying cells, charged particles, nimbus clouds, planetary atmospheres, galactic clusters. We understand that, at least around the edges, some satisfied customers can be somewhat unsatisfied, some A students might equally be B+ students, some stars are as much in a galactic cluster as out of it. Events can more or less smoothly transition to their opposites, making classification hard near the midpoint of the transition. But in theory—in formal descriptions and in textbooks—the events and their opposites are black and white. A hill is a mountain if it is at least x meters tall, not a mountain if it is one micron less than x in height. Every molecule in the universe either is or is not a pencil molecule, even those hovering above the pencil's surface.

Consider some further examples. The probability that this essay gets published is one thing. The degree to which it gets published is another. The essay may be edited in hundreds of ways. Or the essay may be marred with typographical errors, and so on.

Question: Does quantum mechanics deal with the probability that an unambiguous electron occupies spacetime points? Or does it deal with the degree to which an electron, or an electron smear, occurs at spacetime points? Does $|\psi|^2 dV$ measure the probability that a random-point electron occurs in infinitesimal volume dV? Or[12] does it measure the degree to which a deterministic electron cloud occurs in dV? Different interpretation, different universe. Perhaps even existence admits degrees.

Suppose there is 50% chance that there is an apple in the refrigerator (electron in a cell[12]). That is one state of affairs, perhaps arrived at through frequency calculations or a Bayesian state of knowledge. Now suppose there is half an apple in the refrigerator. That is another state of affairs. Both states of affairs are superficially equivalent in terms of their numerical uncertainty. Yet physically, ontologically, they are distinct. One is "random", the other fuzzy.

If events are *assumed* unambiguous, as in balls-in-urns experiments, there is no fuzziness. Only randomness remains. But when discussing the physical universe, every assertion of event ambiguity or nonambiguity is an empirical *hypothesis*. This is habitually overlooked when applying probability theory. Years of such oversight are perhaps responsible for the deeply entrenched sentiment that uncertainty is randomness, and randomness alone. The silent assumption of universal nonambiguity is akin to the pre-relativistic assumption of an uncurved

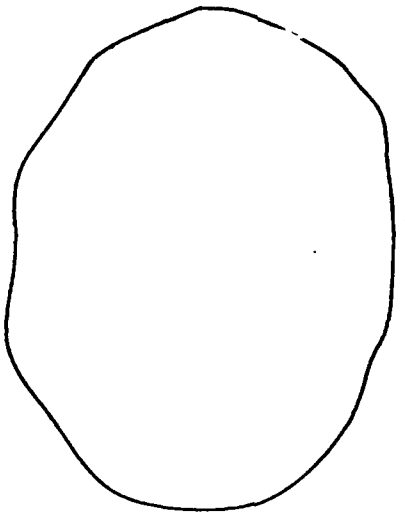


Figure 1 Inexact oval. Which statement better describes the situation: (a) "It is probably an ellipse" or (b) "It is a fuzzy ellipse"?

universe. $A \cap A^c = \emptyset$ is the "parallel postulate" of classical set theory and logic, indeed of Western thought.

If fuzziness is a unique type of uncertainty, if fuzziness exists, the physical consequences are universal, and the sociological consequence is startling: scientists, especially physicists, have overlooked an entire mode of reality.

Fuzziness is a type of deterministic uncertainty. Ambiguity is a property of physical phenomena. Unlike fuzziness, probability dissipates with increasing information. After the fact "randomness" looks like fiction. (This is especially awkward since in general the laws of science are time reversible, invariant if time t is replaced with time $-t$. Where does the randomness go?) Yet there is as much ambiguity after a sample-space experiment as before. Increasing information tends to specify the degrees of occurrence. Even if science had run its course and all the facts were in, a platypus would remain only roughly a mammal, a large hill only roughly a mountain, an oval squiggle only roughly an ellipse. Fuzziness does not require that God play dice.

Consider the inexact oval in Figure 1. Does it make more sense to say that the oval is *probably* a circle (or ellipse), or that it *is* a fuzzy ellipse? There is nothing random about the matter. The situation is deterministic: All the facts are in. Yet uncertainty remains. The uncertainty is due to the simultaneous occurrence of two properties: to some extent the inexact oval is an ellipse and to some extent it is not an ellipse.

More formally, is $m_A(x)$, the degree to which element x belongs to fuzzy set A, simply the probability that x is in A? Is $m_A(x) = \text{Prob}\{x \in A\}$ true? Cardinality-wise, sample spaces cannot be too big. Else a positive measure cannot be both countably additive and finite, and thus a probability measure. The space of all possible oval figures is too big, since there are more of these than real numbers. Almost all sets are too big to define probabilities, yet fuzzy sets can always be defined.

$\text{Prob}\{x \in A\}$ might be interpreted as the probability of a fuzzy event, the probability that element x belongs to fuzzy set A with degree $m_A(x)$. Rarely indeed then should the equality $\text{Prob}\{x \in A\} = m_A(x)$ occur.

But this is not the intended interpretation of the assertion $\text{Prob}\{x \in A\} = m_A(x)$. Instead set A is not fuzzy. The element x either is or is not an element of set A. We do not know which, and we describe this uncertainty with the probability $\text{Prob}\{x \in A\}$. But then surely $\text{Prob}\{x \in A\} \neq m_A(x)$. For example, $\text{Prob}\{x \in A \cap A^c\} = 0$ and $\text{Prob}\{x \in A \cup A^c\} = 1$ for every nonfuzzy set A. Yet $m_{A \cap A^c}(x) > 0$ and $m_{A \cup A^c}(x) < 1$ for every properly fuzzy set A.

Probability theory is a chapter in the book of finite measure theory. Many probabilists do not care for this classification, but they fall back upon it when defining terms.[7] How reasonable is it to believe that finite measure theory—ultimately, the summing of nonnegative numbers to unity—exhaustively describes the (quantum-mechanical) universe? Does it really describe *any* thing?

Surely from time to time every probabilist wonders whether probability describes anything real. From Democritus to Einstein, there has been the suspicion that, as David Hume[5] put it,

> though there be no such thing as *chance* in the world, our ignorance of the real cause of any event has the same influence on the understanding and begets a like species of belief.

When we model noisy processes by extending differential equations to stochastic differential equations, it seems we introduce the formalism only as a working approximation to several underlying unspecified processes, processes that presumably obey deterministic differential equations. In this sense conditional expectations and martingale techniques might seem reasonably applied, for example, to stock options or commodity futures phenomena, where the behavior involved consists of aggregates of aggregates of aggregates. The same techniques seem less reasonably applied to quarks, leptons, and void.

3. THE UNIVERSE AS A FUZZY SET

The world, as Wittgenstein[11] observed, is everything that is the case. In this spirit we can summarize the ontological case for fuzziness: The universe consists of all subsets of the universe. The only subsets of the universe that are not fuzzy are the constructs of classical mathematics. All other sets—sets of particles, cells, tissues, people, ideas, galaxies—in principal contain elements to different degrees. Their membership is partial, graded, inexact, ambiguous, or uncertain.

The same universal circumstance holds at the level of logic and truth. The only logically true or false statements—statements S with truth value $t(S)$ in $\{0, 1\}$—are tautologies, theorems, and contradictions. If S is any statement about the universe, an empirical statement, then $0 < t(S) < 1$ holds by the canons of scientific method and by the lack of a single demonstrated factual statement S with $t(S) = 1$ or $t(S) = 0$. That is the thrust of Einstein's quote above.

Fuzziness arises from the ambiguity between a thing A and its opposite A^c. If we do not know A with certainty, we do not know A^c with certainty either. Else by double negation we would know A with certainty. This produces nondegenerate *overlap*: $A \cap A^c \neq \emptyset$, which breaks the "law of noncontradiction". Equivalently, this also produces nondegenerate *underlap*:[10] $A \cup A^c \neq X$, which breaks the "law of excluded middle". Here X is the ground set or universe of discourse. Recall[4] that these laws are never broken in probabilistic or stochastic logics—$P(A \text{ and not-}A) = 0$ and $P(A \text{ or not-}A) = 1$—even though they are broken with many, perhaps most, human utterances. Nor are probability measures allowed to take such fuzzy

sets as arguments. The sets must first be quantized, rounded off, or defuzzified to the nearest nonfuzzy set. So the question arises: How mathematically natural are fuzzy sets?

4. THE GEOMETRY OF FUZZY SETS: SETS AS POINTS

It helps to see the geometry of fuzzy sets when discussing fuzziness. To date this visual property has been overlooked. The emphasis has instead been on interpreting fuzzy sets as membership functions, mappings m_A from domain X to range $[0, 1]$. But functions are hard to visualize. Membership functions are often pictured as two-dimensional graphs, with the domain X misleadingly represented as one-dimensional. The geometry of fuzzy sets involves both the domain $X = \{x_1, \ldots, x_n\}$ and the range $[0, 1]$ of mappings $m_A: X \to [0, 1]$. The geometry of fuzzy sets is a great aid in understanding fuzziness, defining fuzzy concepts, and proving fuzzy theorems. Visualizing this geometry may by itself be the most powerful argument for fuzziness.

The geometry of fuzzy sets is revealed by asking an odd question: What does the fuzzy power set $F(2^X)$, the set of all fuzzy subsets of X, look like? Answer: A cube. What does a fuzzy set look like? A point in a cube. The set of all fuzzy subsets is the unit hypercube $I^n = [0, 1]^n$. A fuzzy set is any point[11] in the cube I^n. So (X, I^n) is the fundamental measurable space of (finite) fuzzy theory. The theory of fuzzy sets—more accurately, the theory of *continuous* sets—can be taught on a Rubik's cube.

Vertices of the cube I^n are nonfuzzy sets. So the ordinary power set 2^X, the set of all 2^n nonfuzzy subsets of X, is the Boolean n-cube B^n: $2^X = B^n$. Fuzzy sets fill in the lattice B^n to produce the solid cube I^n: $F(2^X) = I^n$.

Consider the set of two elements $X = \{x_1, x_2\}$. The nonfuzzy power set 2^X contains four sets: $2^X = \{\emptyset, X, \{x_1\}, \{x_2\}\}$. These four sets correspond respectively to the four bit vectors (0 0), (1 1), (1 0), and (0 1). The 1s and 0s indicate the presence or absence of the ith element x_i in the subset. More abstractly, each subset A is uniquely defined by one of the two-valued membership functions $m_A: X \to \{0, 1\}$.

Now consider the fuzzy subsets of X. The fuzzy subset $A = (\frac{1}{3}\ \frac{3}{4})$ can be viewed as one of the continuum-many continuous-valued membership functions $m_A: X \to [0, 1]$. Indeed this is the classical Zadeh[16] *sets-as-functions* definition of fuzzy sets. In this example element x_1 belongs to, or fits in, subset A a little bit—to degree $\frac{1}{3}$. Element x_2 has more membership than not at $\frac{3}{4}$. Analogous to the bit vector representation of finite (countable) sets, we say that A is represented by the *fit vector* $(\frac{1}{3}\ \frac{3}{4})$. The element $m_A(x_i)$ is the ith *fit*[10] or *fuzzy unit* value. The set-as-points view then geometrically represents the fuzzy subset A as a point in I^2, the unit square, as in Figure 2.

The midpoint of the cube I^n is maximally fuzzy. All its membership values are $\frac{1}{2}$. The midpoint is unique in two respects. First, the midpoint is the only set A that not only equals its own opposite A^c but equals its own overlap and underlap as well:

$$A = A \cap A^c = A \cup A^c = A^c.$$

Second, the midpoint is the only point in the cube I^n that is equidistant to each

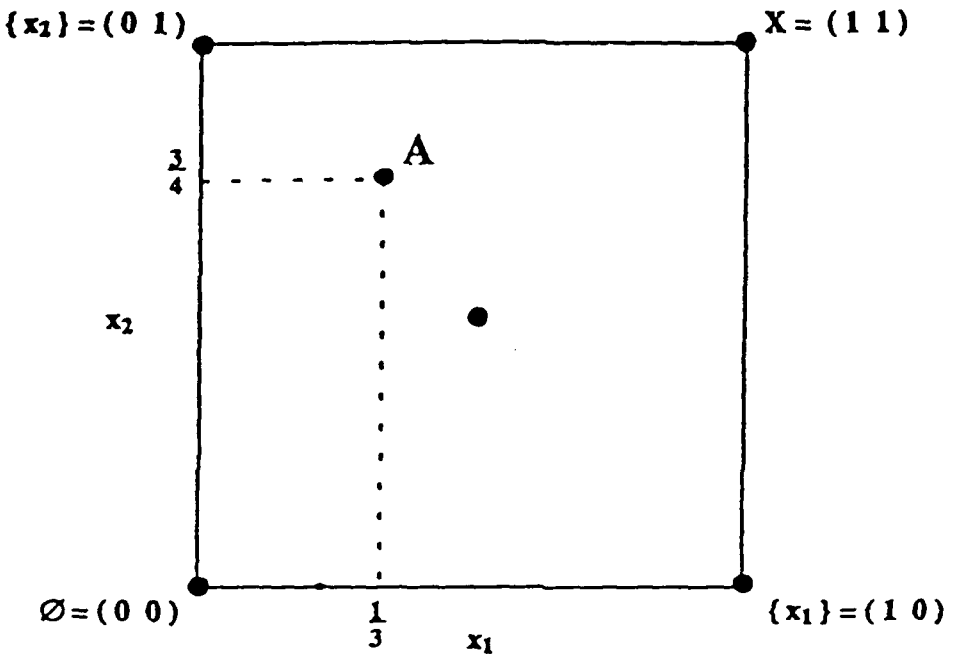


Figure 2 Sets as points. The fuzzy subset A is a point in the unit 2-cube with coordinates or fit values $(\frac{1}{3}\ \frac{3}{4})$. The first element x_1 fits in or belongs to A to degree $\frac{1}{3}$, the element x_2 to degree $\frac{3}{4}$. The cube consists of all possible fuzzy subsets of two elements $\{x_1, x_2\}$. The four corners represent the power set 2^X of $\{x_1, x_2\}$.

of the 2^n vertices of the cube. The nearest corners are also the farthest. This metrical relationship is evident in Figure 2.

Fuzzy sets are combined[16] pairwise with minimum, maximum, and order reversal, as are nonfuzzy sets. Fuzzy set intersection is defined fitwise by pairwise minimum (picking the smaller of the two elements), union by pairwise maximum, and complementation by order reversal. For example:

$$A = (1\ 0.8\ 0.4\ 0.5)$$

$$B = (0.9\ 0.4\ 0\ 0.7)$$

$$A \cap B = (0.9\ 0.4\ 0\ 0.5)$$

$$A \cup B = (1\ 0.8\ 0.4\ 0.7)$$

$$A^c = (0\ 0.2\ 0.6\ 0.5)$$

$$A \cap A^c = (0\ 0.2\ 0.4\ 0.5)$$

$$A \cup A^c = (1\ 0.8\ 0.6\ 0.5).$$

Note that the overlap fit vector $A \cap A^c$ is not the vector of all zeroes and the underlap fit vector $A \cup A^c$ is not the vector of all ones. This is true of all properly fuzzy sets, all points in I^n other than vertex points. Indeed the min-max definitions

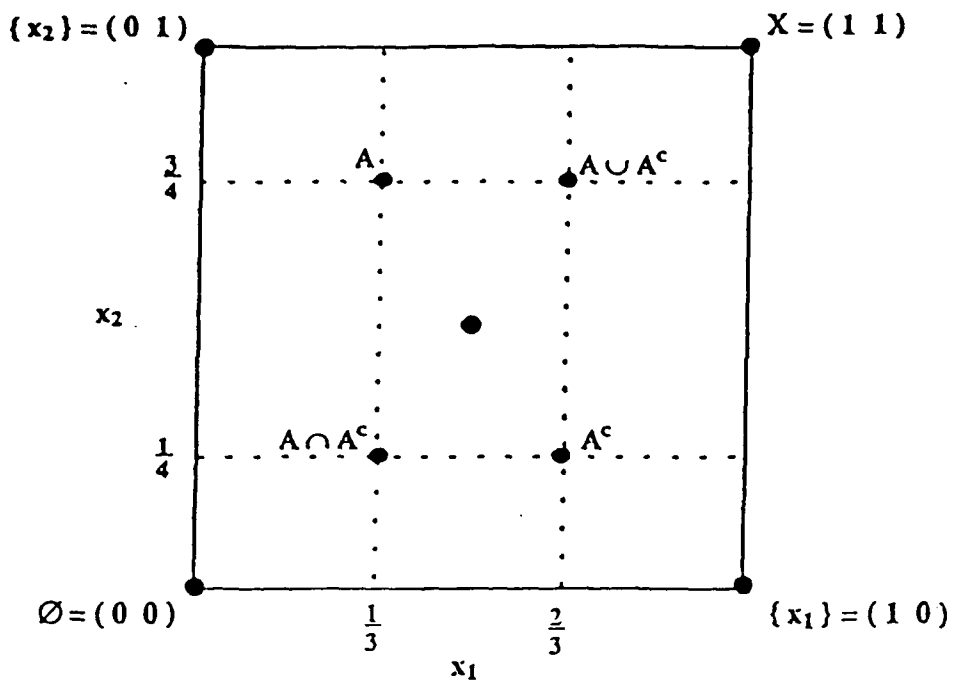


Figure 3 Completing the fuzzy square. The fuzzier A is, the closer A is to the midpoint of the fuzzy cube. As A approaches the midpoint, all four points—A, A^c, $A \cap A^c$, and $A \cup A^c$—contract to the midpoint. The less fuzzy A is, the closer A is to the nearest vertex. As A approaches the vertex, all four points spread out to the four vertices and the bivalent power set 2^X is recovered.

give at once the following fundamental characterization of fuzziness as non-degenerate overlap and nonexhaustive underlap.

PROPOSITION *A is properly fuzzy if and only if $A \cap A^c \neq \emptyset$ and if and only if $A \cup A^c \neq X$.*

An illustration of this fundamental proposition is what we might call *completing the fuzzy square.* Consider again the two-dimensional fuzzy set A defined by the fit vector $(\frac{1}{3}\ \frac{3}{4})$. The corresponding overlap and underlap sets can be found by first finding the complement set A^c and then combining the fit vectors pairwise with minimum and with maximum:

$$A = (\tfrac{1}{3}\ \tfrac{3}{4})$$

$$A^c = (\tfrac{2}{3}\ \tfrac{1}{4})$$

$$A \cap A^c = (\tfrac{1}{3}\ \tfrac{1}{4})$$

$$A \cup A^c = (\tfrac{2}{3}\ \tfrac{3}{4}).$$

The sets-as-points view shows that these four points in the unit square hang together, indeed move together, in a very natural way. Consider the geometry of Figure 3.

In Figure 3 the four fuzzy sets involved in the fuzziness of set A—the sets A, A^c, $A \cap A^c$, and $A \cup A^c$—contract to the midpoint as A becomes maximally fuzzy and

expand out to the Boolean corners of the cube as A becomes minimally fuzzy. The same contraction and expansion occurs in n dimensions for the 2^n fuzzy sets defined by all combinations of $m_A(x_1)$ and $m_{A^c}(x_1), \ldots, m_A(x_n)$ and $m_{A^c}(x_n)$.

At the midpoint nothing is distinguishable. At the vertices everything is distinguishable. These extremes represent the two ends of the spectrum of logic and set theory. In this sense the midpoint is the black hole of set theory.

5. PARADOX AT THE MIDPOINT

The midpoint is full of paradox. It is forbidden to classical logic and set theory. Where midpoint phenomena appear in Western thought they are invariably labeled "paradoxes" or denied altogether. Midpoint phenomena include the half-empty and half-full cup, the Taoist Yin-Yang, the liar from Crete who said that all Cretans are liars, Bertrand Russell's set of all sets that are not members of themselves, and Russell's barber.

Russell's barber is a bewhiskered man who lives in a town and shaves a man if and only if he does not shave himself. So who shaves the barber? If he shaves himself, then by definition he does not. But if he does not shave himself, then by definition he does. So he does and he does not—contradiction ("paradox"). Gaines[4] observed that this paradoxical circumstance can be numerically interpreted as follows.

Let S be the proposition that the barber shaves himself and not-S that he does not. Then since S implies not-S and not-S implies S, the two propositions are logically equivalent: $S = \text{not-}S$. Equivalent propositions have the same truth values:

$$t(S) = t(\text{not-}S)$$

$$= 1 - t(S).$$

Solving for $t(S)$ gives the midpoint point of the truth interval (the one-dimensional cube $[0, 1]$): $t(S) = \frac{1}{2}$. The midpoint is equidistant to the vertices 0 and 1. In the bivalent (two-valued) case, roundoff is impossible and paradox occurs.

In bivalent logic both statements S and not-S must have truth value zero or unity. The fuzzy resolution of the paradox only uses the fact that the truth values are equal. It does not in principle constrain their range. The midpoint value $\frac{1}{2}$ emerges from the structure of the problem and the order-reversing effect of negation.

The paradoxes of classical set theory and logic are part of the price one pays for an arbitrary insistence on bivalence. This insistence is often made in the name of science. In the end, though, if is simply a cultural preference, a reflection of an educational predilection that goes back at least to Aristotle. It takes great faith to insist on bivalence in the face of both bivalent contradictions (paradoxes) and a consistent fuzzy alternative.

Put another way, fuzziness shows that there are limits to logical certainty. We can no longer assert the laws of noncontradiction and excluded middle *for sure*—and *for free*.

The fuzzy theorist must explain why so many people have been wrong for so long. We now have the machinery to offer an explanation. The reason is that *rounding off*, quantizing, simplifies life and often costs little. We agree to call empty

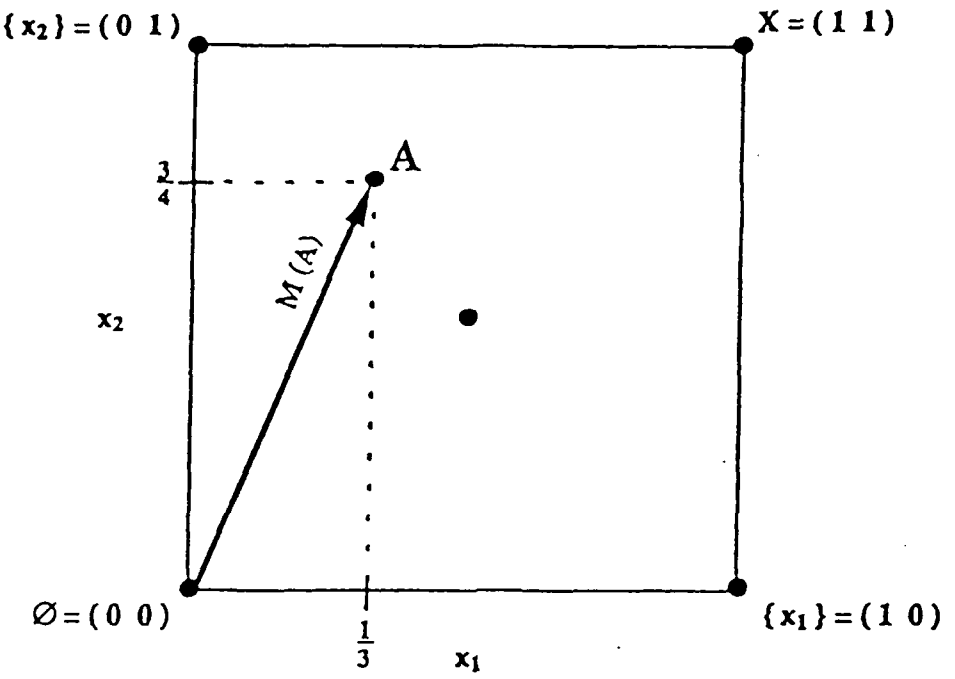


Figure 4 The count $M(A)$ of A is the fuzzy Hamming norm (l^1 norm) of the vector drawn from the origin to A.

the near empty cup, and present the large pulse and absent the small pulse. We round off points inside the fuzzy cube to the nearest vertex. This roundoff heuristic works fine as a first approximation to describing the universe until we get near the midpoint of the cube. These phenomena are harder to roundoff. In the logically extreme case, at the midpoint of the cube, the procedure breaks down completely because every vertex is equally close. Hands are thrown up and paradox declared.

Faced with midpoint phenomena, the fuzzy skeptic is in the same position as the flat-earther, who denies that the earth's surface is curved, when she stands at the north pole, looks at her compass and wants to go south.

6. COUNTING WITH FUZZY SETS

How big is a fuzzy set? The size or cardinality of A, $M(A)$, is the sum of the fit values of A:

$$M(A) = \sum_{i=1}^{n} m_A(x_i).$$

The count of $A = (\frac{1}{3}\ \frac{3}{4})$ is $M(A) = \frac{1}{3} + \frac{3}{4} = \frac{13}{12}$. The cardinality measure M is sometimes called the *sigma-count*.[17] The measure M generalizes[9] the classical counting measure of combinatorics and measure theory. (So (X, I^n, M) is the fundamental measure space of fuzzy theory.) In general the measure M does not give integer values.

The measure M has a natural geometric interpretation in the sets-as-points framework. It is the magnitude of the vector drawn from the origin to the fuzzy set, as illustrated in Figure 4.

Consider the l^p distance between fuzzy sets A and B in I^n:

$$l^p(A, B) = \sqrt[p]{\sum_{i=1}^{n} |m_A(x_i) - m_B(x_i)|^p},$$

where $1 \leqq p \leqq \infty$. The l^2 distance is the physical Euclidean distance actually illustrated in the figures. The simplest distance is the l^1 or *fuzzy Hamming distance*, the sum of the absolute fit differences. We shall use fuzzy Hamming distance throughout, though all results admit a general l^p formulation. Using the fuzzy Hamming distance the count M can be rewritten as the desired l^1 norm:

$$\begin{aligned} M(A) &= \sum_{i=1}^{n} m_A(x_i) \\ &= \sum_i |m_A(x_i) - 0| \\ &= \sum_i |m_A(x_i) - m_{\varnothing}(x_i)| \\ &= l^1(A, \varnothing). \end{aligned}$$

7. THE FUZZY ENTROPY THEOREM

How fuzzy is a fuzzy set? Fuzziness is measured by a *fuzzy entropy* measure. Entropy is a generic notion. It need not be probabilistic. Entropy measures the uncertainty of a system or message. A fuzzy set is a type of system or message. Its uncertainty is its fuzziness.

The fuzzy entropy of A, $E(A)$, varies from 0 to 1 on the unit hypercube I^n. Only the cube vertices have zero entropy, since nonfuzzy sets are unambiguous. The cube midpoint uniquely has maximum entropy one. Fuzzy entropy smoothly increases as one moves from any vertex to the midpoint. The algebraic requirements for fuzzy entropy measures can be found in Klir.[8]

Simple geometric considerations lead[10] to a ratio form for the fuzzy entropy. The closer the fuzzy set A is to the nearest vertex A_{near}, the farther A is from the farthest vertex A_{far}. Opposite the long diagonal from the nearest vertex is the farthest vertex. Let a denote the distance $l^1(A, A_{near})$ to the nearest vertex and let b denote the distance $l^1(A, A_{far})$ to the farthest vertex. Then the fuzzy entropy is simply the ratio of a to b:

$$E(A) = \frac{a}{b} = \frac{l^1(A, A_{near})}{l^1(A, A_{far})}.$$

The sets-as-points interpretation of the fuzzy entropy is shown in Figure 5, where $A = (\frac{1}{3}\ \frac{3}{4})$, $A_{near} = (0\ 1)$, and $A_{far} = (1\ 0)$. So $a = \frac{1}{3} + \frac{1}{4} = \frac{7}{12}$ and $b = \frac{2}{3} + \frac{3}{4} = \frac{17}{12}$. So $E(A) = \frac{7}{17}$.

Alternatively, those reading this in a room can imagine that the room is the unit

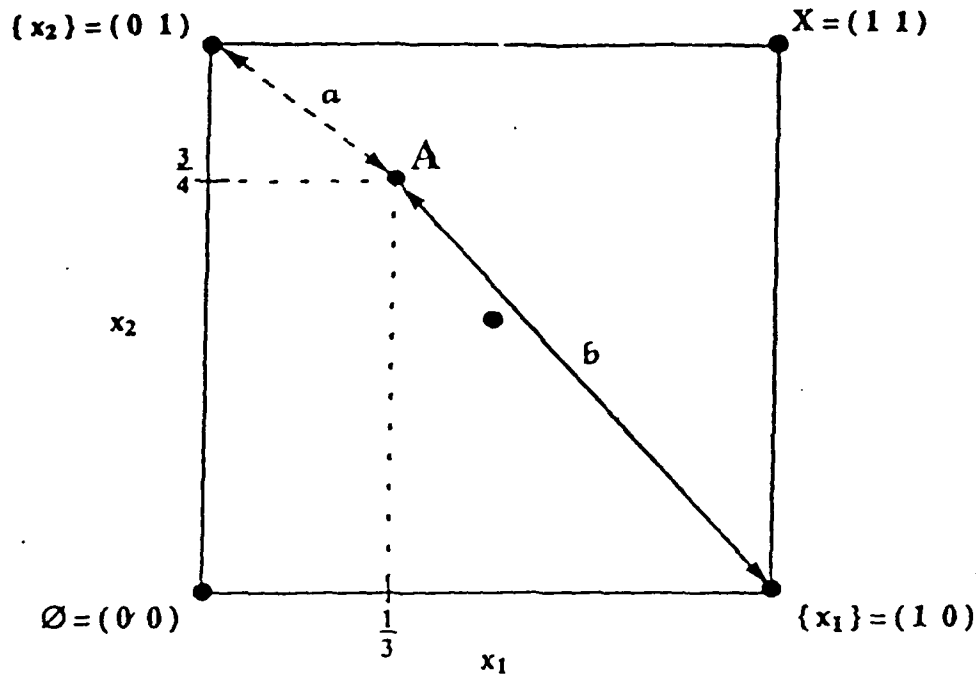


Figure 5 Fuzzy entropy $E(A)=a/b$ as balance between distance to nearest vertex and distance to farthest vertex.

cube I^3 and that their head is a fuzzy set in it. Once the nearest corner of the room is located, the farthest corner is opposite the long diagonal emanating from the nearest corner. If your head is in a corner, $a=0$ and $E(A)=0$. If your head is in the metrical center of the room, every corner is nearest and farthest. So $a=b$ and $E(A)=1$.

Since overlap and underlap characterize fuzziness we can expect them to be involved in the measure of fuzziness. Careful examination of the completed fuzzy square in Figure 3 shows that this is the case. For, by symmetry, each of the four points A, A^c, $A\cap A^c$, and $A\cup A^c$ is equally close to its nearest vertex. The common distance is a. Similarly, each point is equally far from its farthest vertex. The common distance is b. One of the first four distances is the count $M(A\cap A^c)$. One of the second four distances is the count $M(A\cup A^c)$. This gives a geometric proof of the Fuzzy Entropy Theorem,[10,11] which states that fuzziness consists of a balance of counted violations of the law of noncontradiction and counted violations of the law of excluded middle.

FUZZY ENTROPY THEOREM

$$E(A)=\frac{M(A\cap A^c)}{M(A\cup A^c)}.$$

An algebraic proof is straightforward. The geometric proof can be seen by examining the completed fuzzy square in Figure 6.

The Fuzzy Entropy Theorem explains why fuzziness begins where Western logic ends. When the laws of noncontradiction and excluded middle are obeyed, overlap is empty and underlap is exhaustive. So $M(A\cap A^c)=0$ and $M(A\cup A^c)=n$, and thus $E(A)=0$.

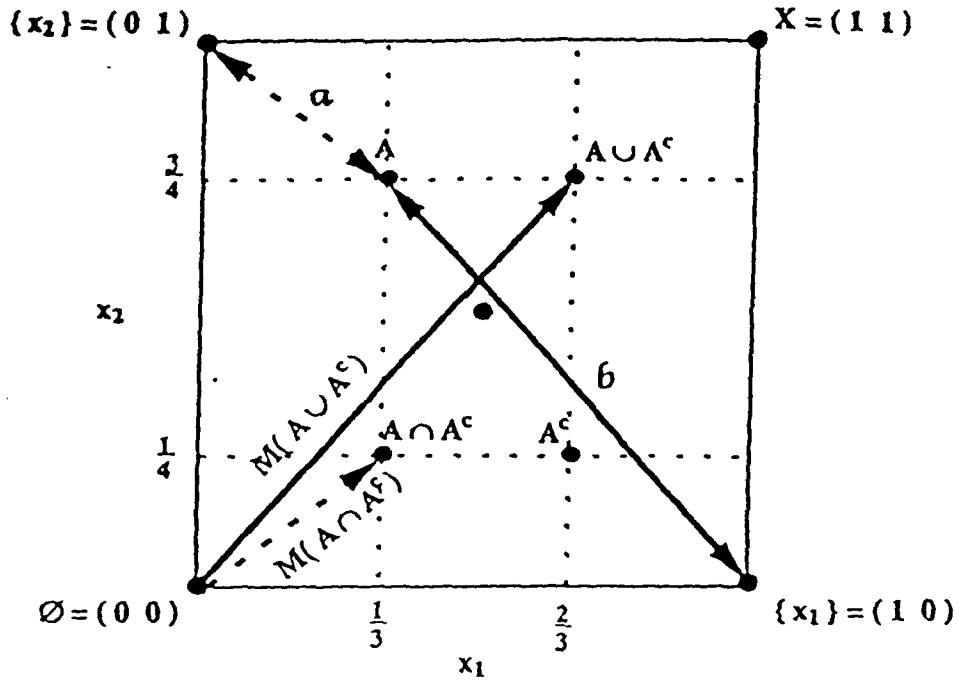


Figure 6 Geometry of the Fuzzy Entropy Theorem. By symmetry each of the four points on the completed fuzzy square is equally close to its nearest vertex and equally far from its farthest vertex.

The Fuzzy Entropy Theorem also provides a first-principles derivation of the basic fuzzy set operations of minimum (intersection), maximum (union), and order reversal (complementation) proposed in 1965 by Zadeh[16] at the inception of fuzzy theory. (Lukasiewicz first proposed these operations for continuous or fuzzy logics in the 1920s.)

For the fuzzy theorist, this result also shows that triangular norms or T-norms,[8] which generalize conjunction or intersection, and the dual triangular co-norms C, which generalize disjunction or union, do not have the first-principles status of min and max. For, the triangular norm inequalities,

$$T(x, y) \leqq \min(x, y) \leqq \max(x, y) \leqq C(x, y),$$

show that replacing min with any T in the numerator term $M(A \cap A^c)$ can only make the numerator smaller. Replacing max with any C in the term $M(A \cup A^c)$ can only make the denominator larger. So any T or C not identically min or max makes the ratio smaller, strictly smaller if A is fuzzy. Then the entropy theorem does not hold and the resulting pseudo-entropy measure does not equal unity at the midpoint, though it continues to be maximized there. This can be easily seen with the product T-norm[14] $T(x, y) = xy$ and its DeMorgan dual co-norm $C(x, y) = 1 - T(1-x, 1-y) = x + y - xy$, or with the bounded sum T-norm $T(x, y) = \max(0, x + y - 1)$ and DeMorgan dual $C(x, y) = \min(1, x + y)$. The Entropy Theorem similarly fails in general if the negation or complementation operator $N(x) = 1 - x$ is replaced by a parameterized operator $N_a(x) = (1 - x)/(1 + ax)$ for nonzero $a > -1$.

As an aside, note that all probability distributions, all sets A with $M(A) = 1$, in I^n form a $n-1$ dimensional simplex S^n. In the unit square the probability simplex is

the negatively sloped diagonal line. In the unit 3-cube it is a solid triangle. In the unit 4-cube it is a tetrahedron, and so on up.

If no probabilistic fit value p_i is such that $p_i > \frac{1}{2}$, then the Fuzzy Entropy Theorem implies[11] that the distribution P has fuzzy entropy $E(P) = 1/(n-1)$. Else $E(P) < 1/(n-1)$. So the probability simplex S^n is entropically degenerate for large dimensions n. This result also shows that the uniform distribution $(1/n, \ldots, 1/n)$ maximizes fuzzy entropy on S^n but not uniquely. This in turn shows that fuzzy entropy differs from the average-information measure of probabilistic entropy, which is uniquely maximized by the uniform distribution.

The Fuzzy Entropy Theorem also implies that, analogous to $\log 1/p$, a unit of fuzzy information is $f/(1-f)$ or $(1-f)/f$, depending on whether the fit value f obeys $f \leqq \frac{1}{2}$ or $f \geqq \frac{1}{2}$.

The event x can be ambiguous or clear. It is ambiguous if f is approximately $\frac{1}{2}$ and clear if f is approximately 1 or 0. If an ambiguous event occurs, is observed, is disambiguated, etc., then it is maximally informative: $E(f) = E(\frac{1}{2}) = 1$. If a clear event occurs, is observed, etc., it is minimally informative: $E(f) = E(0) = E(1) = 0$. This is in accord with the information interpretation of the probabilistic entropy measure $\log 1/p$, where the occurrence of a sure event ($p = 1$) is minimally informative (zero entropy) and the occurrence of an impossible event ($p = 0$) is maximally informative (infinite entropy).

8. THE SUBSETHOOD THEOREM

Sets contain subsets. A is a *subset* of B, denoted $A \subset B$, if and only if every element of A is an element of B. The power set 2^B contains all of B's subsets. So, alternatively,[1] A is a subset of B just in case A belongs to B's power set:

$$A \subset B \quad \text{if and only if } A \in 2^B.$$

The subset relation corresponds to the implication relation in logic. In classical logic *truth* is a mapping from the set of statements $\{S\}$ to truth values: $t: \{S\} \to \{0, 1\}$. Consider the truth-tabular definition of implication for bivalent propositions P and Q:

P	Q	$P \to Q$
0	0	1
0	1	1
1	0	0
1	1	1

The implication is false if and only if the antecedent P is true and the consequent Q is false—when "truth implies falsehood".

The same holds for subsets. Representing sets as bivalent functions $m_A: X \to \{0, 1\}$, A is a subset of B if there is no element x that belongs to A but not to B, or $m_A(x) = 1$ but $m_B(x) = 0$. This membership-function definition can be rewritten as follows:

$$A \subset B \quad \text{if and only if } m_A(x) \leqq m_B(x) \quad \text{for all } x.$$

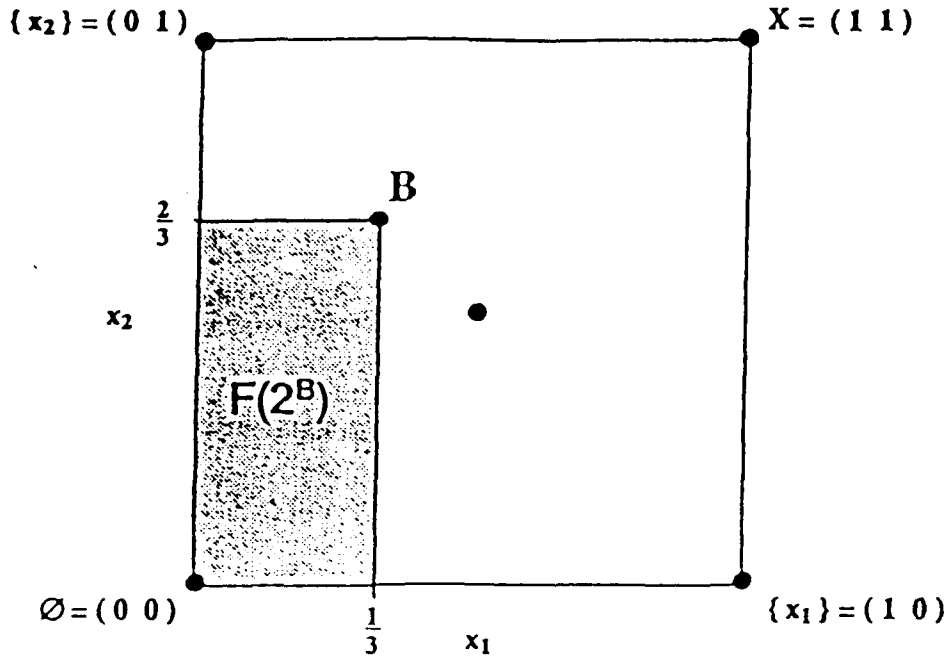


Figure 7 Fuzzy power set $F(2^B)$ as a hyper-rectangle in the fuzzy cube. Side lengths are the fit values $m_B(x_i)$. The size or volume of $F(2^B)$ is the product of the fit values.

Zadeh[16] proposed the same relation for defining when fuzzy set A is a subset of fuzzy set B. We refer to this as the *dominated membership function relationship*. If $A=(0.3\,0\,0.7)$ and $B=(0.4\,0.7\,0.9)$, then A is a fuzzy subset of B but B is not a fuzzy subset of A. A candidate fuzzy set A either is or is not a fuzzy subset of B. This is the problem. The relation of fuzzy subsethood is *not* fuzzy. It is either black or white.

The sets-as-points view asks a geometric question: What do all fuzzy subsets of B look like? What does the *fuzzy power set* of B—$F(2^B)$, the set of all fuzzy subsets of B—look like? The dominated membership function relationship implies that $F(2^B)$ is the hyper-rectangle emanating from the origin with side lengths given by the fit values $m_A(x_i)$. Figure 7 displays the fuzzy power set of the set $B=(\frac{1}{3}\,\frac{2}{3})$. Of course the count of $F(2^B)$ is infinite if B is not empty. For finite-dimensional sets, the size of $F(2^B)$ can be taken[11] as the Lebesgue measure or volume $V(B)$, the product of the fit values:

$$V(B)=\prod_{i=1}^{n} m_B(x_i).$$

Figure 7 illustrates that $F(2^B)$ is not a fuzzy set. A cube point A either is or is not in the hyper-rectangle $F(2^B)$. Some points A outside the hyper-rectangle $F(2^B)$ resemble subsets of B more than other points do. The black–white definition of subsethood ignores this.

The natural generalization is to define fuzzy subsets on $F(2^B)$. Some sets A belong in $F(2^B)$ to different degrees. The abstract membership function $m_{F(2^B)}(A)$ can be any number in $[0, 1]$. Degrees of subsethood are possible.

Let $S(A, B)$ denote the degree to which A is a subset of B:

$$S(A, B) = \text{Degree}(A \subset B)$$

$$= m_{F(2^n)}(A).$$

$S(\cdot, \cdot)$ is the *subsethood measure*. $S(\cdot, \cdot)$ takes values in $[0, 1]$. We will see that it is the fundamental, unifying structure in fuzzy theory.

The current task is to measure $S(A, B)$. We will first present an earlier[10,11] algebraic derivation of the subsethood measure $S(A, B)$. We will then present a new, more fundamental, geometric derivation.

The algebraic derivation is the *fit-violation strategy*.[10] The idea is that you study a law by breaking it. Consider the dominated membership function relationship: $A \subset B$ if and only if $m_A(x) \leqq m_B(x)$ for all x in X.

Suppose element x_v violates the dominated membership function relationship: $m_A(x_v) > m_B(x_v)$. Then A is not a subset of B, at least not totally. Suppose further that the dominated membership inequality holds for all other elements x. Only element x_v violates the relationship. For instance, X may consist of one hundred values: $X = \{x_1, \ldots, x_{100}\}$. The violation might occur, say, with the first element: $x_1 = x_v$. Then intuitively A is largely a subset of B. Suppose now that X contains a thousand elements, or a trillion elements, and only the first element violates the dominated membership function relationship. Then surely A is overwhelmingly a subset of B; perhaps $S(A, B) = 0.999999999999$.

This argument suggests we should count fit violations in magnitude and frequency. The greater the violations in magnitude, $m_A(x_v) - m_B(x_v)$, and the greater the number of violations relative to the size $M(A)$ of A, the less A is a subset of B; equivalently, the more A is a *superset* of B. For, both intuitively and by the dominated-membership definition, supersethood and subsethood are inversely related:

$$SUPERSETHOOD(A, B) = 1 - S(A, B).$$

The simplest way to count violations is to add them. If we sum over all x, the summand should equal $m_A(x_v) - m_B(x_v)$ when this difference is positive, zero when it is nonpositive. So the summand is $\max(0, m_A(x) - m_B(x))$. The unnormalized count is therefore the sum of these maxima:

$$\sum_{x \in X} \max(0, m_A(x) - m_B(x)).$$

The simplest, and most appropriate, normalization factor is the count of A, $M(A)$. We can assume $M(A) > 0$ since $M(A) = 0$ if and only if A is empty. The empty set trivially satisfies the dominated membership function relationship. So it is a subset of every set. Normalization gives the minimal measure of nonsubsethood, of supersethood:

$$SUPERSETHOOD(A, B) = \frac{\sum_x \max(0, m_A(x) - m_B(x))}{M(A)}.$$

Then subsethood is the negation of this ratio. This gives the minimal fit-violation measure of subsethood:

$$S(A,B)=1-\frac{\sum_x \max(0, m_A(x)-m_B(x))}{M(A)}.$$

The subsethood measure may appear ungraceful at first but it behaves as it should. Observe that $S(A,B)=1$ if and only if the dominated membership function relationship holds. For if it holds, zero violations are summed. Then $S(A,B)=1-0=1$. If $S(A,B)=1$, every numerator summand is zero. So no violation occurs. At the other exteme, $S(A,B)=0$ if and only if B is the empty set. So the empty set is the unique set without subsets, fuzzy or nonfuzzy. Degrees of subsethood occur between these extremes.

The subsethood measure also relates to logical implication. Viewed at the 1-dimensional level of fuzzy logic, and so ignoring the normalizing count ($M(A)=1$), the subsethood measure reduces to the Lukasiewicz implication operator:

$$\begin{aligned} S(A,B) &= 1-\max(0, m_A-m_B) \\ &= 1-[1-\min(1-0, 1-(m_A-m_B))] \\ &= \min(1, 1-m_A+m_B) \\ &= t_L(A\rightarrow B), \end{aligned}$$

which clearly generalizes the truth-tabular definition of bivalent implication.

Consider the fit vectors $A=(0.2\,0\,0.4\,0.5)$ and $B=(0.7\,0.6\,0.3\,0.7)$. Neither set is a proper subset of the other. A is almost a subset of B but not quite since $m_A(x_3)-m_B(x_3)=0.4-0.3=0.1>0$. Hence $S(A,B)=1-\frac{0.1}{1.1}=\frac{10}{11}$. Similarly $S(B,A)=1-\frac{1.3}{2.3}=\frac{10}{23}$.

The concept of subsethood applies to nonfuzzy sets. Consider the sets

$$C=\{x_1, x_2, x_3, x_5, x_7, x_9, x_{10}, x_{12}, x_{14}\}$$

and

$$D=\{x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{12}, x_{13}, x_{14}\}$$

with corresponding bit vectors

$$C=(1\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1)$$

$$D=(0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 1)$$

C and D are not subsets of each other. But C should very nearly be a subset of D since only x_1 violates the dominated membership function relationship. We find $S(C,D)=1-\frac{1}{9}=\frac{8}{9}$ while $S(D,C)=1-\frac{4}{12}=\frac{2}{3}$. So D is more a subset of C than it is not. This is because the two sets are largely equivalent. They have much overlap: $M(C\cap D)=8$. This observation motivates the Fuzzy Subsethood Theorem presented below. First, though, we present a new geometric derivation of the subsethood measure.

Consider the sets-as-points geometry of subsethood in Figure 7. Set A is either

in the hyper-rectangle $F(2^B)$ or not. Intuitively the subsethood of A in B should be nearly unity when A is arbitrarily close to the fuzzy power set $F(2^B)$. The farther away, the less the subsethood $S(A,B)$ or, equivalently, the greater the supersethood.

So the key idea is metrical: How close is A to $F(2^B)$? Let $d(A,F(2^B))$ denote this l^p distance. There is a distance $d(A,B')$ between A and every point B' in the hyper-rectangle, every subset B' of B. The distance $d(A,F(2^B))$ is the smallest such distance. Since the hyper-rectangle $F(2^B)$ is geometrically well behaved—$F(2^B)$ is closed and bounded (compact) and convex—some subset B^* of B achieves this minimum distance. So the infimum, the greatest lower bound, is the distance $d(A,B^*)$:

$$d(A,F(2^B)) = \inf\{d(A,B'): B' \in F(2^B)\}$$

$$= d(A,B^*).$$

The closest set B^* is easy to locate geometrically. In the Euclidean or ℓ^2 case, this is formally due to the geometric Hahn–Banach Theorem since $F(2^B)$ is convex. If A is a subset of B— if A is in the hyper-rectangle $F(2^B)$—then A itself is the closest subset: $A = B^*$. So suppose A is not a proper subset of B.

The unit cube I^n can be sliced into 2^n hyper-rectangles by extending the sides of $F(2^B)$ to hyperplanes. The hyperplanes intersect perpendicularly (orthogonally), at least in the Euclidean case. $F(2^B)$ is one of the hyper-rectangles. The hyper-rectangle interiors correspond to the 2^n cases whether $m_A(x_i) < m_B(x_i)$ or $m_A(x_i) > m_B(x_i)$ for fixed B and arbitrary A. The edges are the loci of points when some $m_A(x_i) = m_B(x_i)$.

The 2^n hyper-rectangles can be classified as *mixed* or *pure* membership domination. In the pure case either $m_A < m_B$ or $m_A > m_B$ holds in the hyper-rectangle interior for all x and all interior points A. In the mixed case $m_A(x_i) < m_B(x_i)$ holds for some of the coordinates x_i and $m_A(x_j) > m_B(x_j)$ holds for the remaining coordinates x_j in the interior for all interior A. So there are only two pure membership-domination hyper-rectangles, the set of proper subsets $F(2^B)$ and the set of proper supersets.

Figure 8 illustrates how the fuzzy power set $F(2^B)$ of $B = (\frac{1}{3}\ \frac{2}{3})$ can be linearly extended to partition the unit square into $2^2 = 4$ rectangles. The non-subsets A_1, A_2 and A_3 reside in distinct quadrants. The northwest and southeast quadrants are the mixed membership-domination rectangles. The southwest and the northeast quadrants are the pure rectangles.

The nearest set B^* to A in the pure superset hyper-rectangle is B itself. The nearest set B^* in the mixed case is found by drawing a perpendicular (orthogonal) line segment from A to $F(2^B)$. Convexity of $F(2^B)$ is responsible. In Figure 8 the perpendicular lines from A_1 and A_3 intersect line edges (1-dimensional linear subspaces) of the rectangle $F(2^B)$. The line from A_2 to B, the corner of $F(2^B)$, is degenerately perpendicular since B is a zero-dimensional linear subspace.

These "orthogonality" conditions are more pronounced in three dimensions. Let your room again be the unit 3-cube. Consider a large dictionary fit snugly against the floor corner corresponding to the origin. Point B is the dictionary corner farthest from the origin. Extending the three exposed faces of the dictionary partitions the room into 8 octants, one of which is occupied by the dictionary.

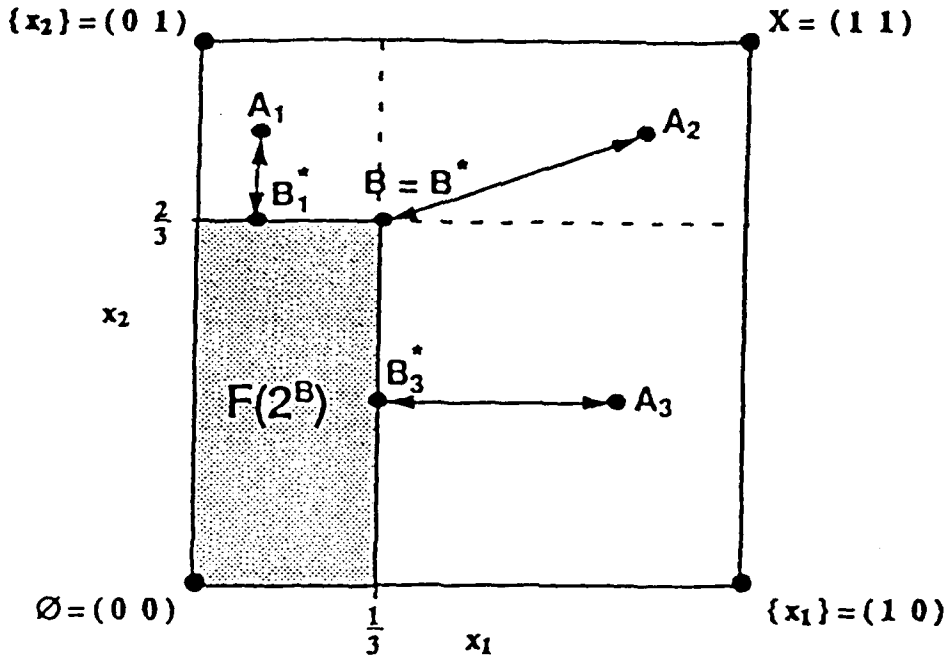


Figure 8 Partition of hypercube I^n into 2^n hyper-rectangles by linearly extending $F(2^B)$. The nearest points B_1^* and B_3^* to points A_1 and A_3 in the northwest and southeast quadrants are found by the normals from $F(2^B)$ to A_1 and A_3. The nearest point B^* to point A_2 in the northeast quadrant is itself. This "orthogonal" optimality condition allows $d(A, B)$ to be given by the general Pythagorean Theorem as the hypotenuse in an ℓ^p "right" triangle.

Points in the other 7 octants are connected to the nearest points on the dictionary by lines that perpendicularly intersect one of the three exposed faces, one of the three exposed edges, or the corner B.

The "orthogonality" condition invokes the ℓ^p-version of the Pythagorean Theorem. For our ℓ^1 purposes:

$$d(A, B) = d(A, B^*) + d(B, B^*).$$

The more familiar l^2-version, actually pictured in Figure 8, requires squaring these distances. For the general ℓ^p case:

$$\|A - B\|^p = \|A - B^*\|^p + \|B^* - B\|^p,$$

or equivalently,

$$\sum_{i=1}^{n} |a_i - b_i|^p = \sum_{i=1}^{n} |a_i - b_i^*|^p + \sum_{i=1}^{n} |b_i^* - b_i|^p.$$

Equality holds for all $p \geq 1$ since, as is clear from Figure 8 and in general, from the algebraic argument below, either $b_i^* = a_i$ or $b_i^* = b_i$.

This Pythagorean equality is surprising. We have come to think of the Pythagorean Theorem (and orthogonality) as an ℓ^2 or Hilbert space property. Yet here it holds in *every* ℓ^p space—if B^* is the set in $F(2^B)$ closest to A in ℓ^p distance. Of course for other sets strict inequality holds in general if $p \neq 2$. This suggests a

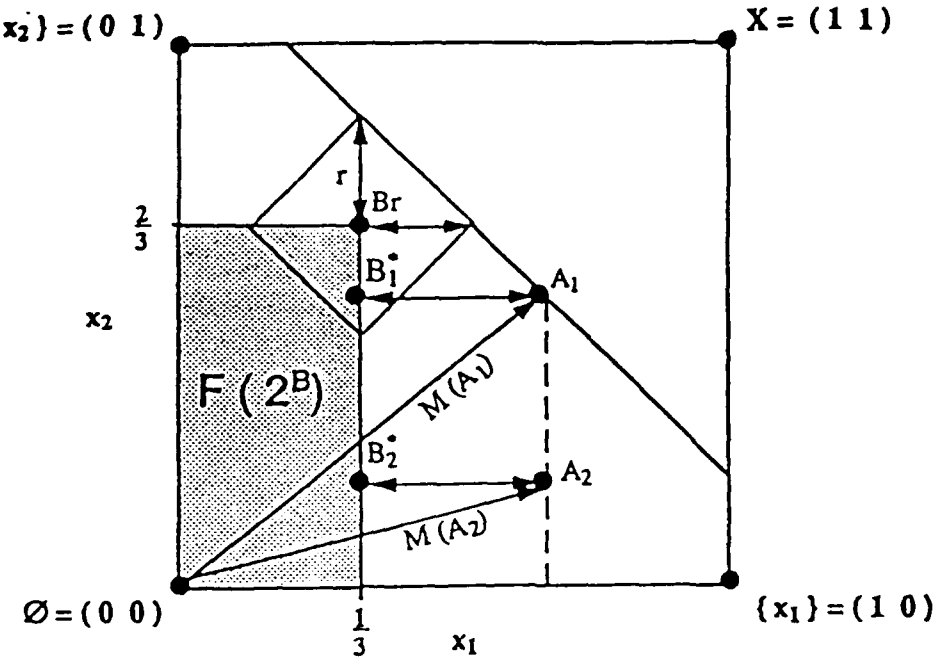


Figure 9 Dependence of subsethood on the count $M(A)$. A_1 and A_2 are equidistant to $F(2^B)$ but A_1 is closer to B than A_2 is; correspondingly, $M(A_1) > M(A_2)$. Loci of points A of constant count $M(A)$ are line segments parallel to the negatively sloping long diagonal. l^1 spheres centered at B are diamond shaped.

special status for the closest set B^*. We shall see below that the Subsethood Theorem confirms this suggestion. We shall use the term "orthogonality" loosely to refer to this ℓ^p Pythagorean relationship, while remembering its customary restriction to ℓ^2 spaces and inner products.

The natural suggestion is to define supersethood as the distance $d(A, F(2^B)) = d(A, B^*)$. Supersethood increases with this distance, subsethood decreases with it. To keep supersethood, and thus subsethood, unit-interval valued, the distance must be suitably normalized.

The simplest way to normalize $d(A, B^*)$ is with a constant: the maximum unit-cube distance, $n^{1/p}$ in the general ℓ^p case and n in our case. This gives the candidate subsethood measure

$$S(A, B) = 1 - \frac{d(A, B^*)}{n}.$$

This candidate subsethood measure fails in the boundary case when B is the empty set. For then $d(A, B^*) = d(A, B) = M(A)$. So the measure gives $S(A, \emptyset) = 1 - (M(A)/n) > 0$. Equality holds exactly when $A = X$. But the empty set has no subsets. The only normalization factor that ensures this is the count $M(A)$. Of course $M(A) = n$ when $A = X$.

Normalizing by n also treats all equidistant points the same. Consider points A_1 and A_2 in Figure 9. Both points are equidistant to their nearest $F(2^B)$ point: $d(A_1, B_1^*) = d(A_2, B_2^*)$. But A_1 is closer to B than A_2. In particular A_1 is closer to the horizontal line defined by the fit value $m_B(x_2) = \frac{2}{3}$. The variable quantity that

reflects this is the count $M(A)$: $M(A_1) > M(A_2)$. The count gap $M(A_1) - M(A_2)$ is due to the fit gap involving x_1, and reflects $d(A_1, B) < d(A_2, B)$. In general the count $M(A)$ relates to this distance, as can be seen by checking extreme cases of closeness of A to B (and drawing some diamond-shaped l^1 spheres centered at B). Indeed if $m_A > m_B$ everywhere, $d(A, B) = M(A) - M(B)$.

Since $F(2^B)$ fits snugly against the origin, the count $M(A)$ in any of the other $2^n - 1$ hyper-rectangles can only be larger than the count $M(B^*)$ of the nearest $F(2^B)$ points. The normalization choice of n leaves the candidate subsethood measure indifferent to which of the $2^n - 1$ hyper-rectangles A is in and to where A is in the hyper-rectangle. Each point in each hyper-rectangle involves a different combination of fit violations and satisfactions. The normalization choice of $M(A)$ reflects this fit violation structure as well as behaves appropriately in boundary cases.

The normalization choice $M(A)$ leads to the subsethood measure

$$S(A, B) = 1 - \frac{d(A, B^*)}{M(A)}.$$

We now show that this measure is equal to the subsethood measure derived algebraically above.

Let B' be any subset of B. Then by definition the nearest subset B^* obeys the inequality:

$$\sqrt[p]{\sum_{i=1}^{n} |a_i - b_i^*|^p} \leqq \sqrt[p]{\sum_{i=1}^{n} |a_i - b_i'|^p},$$

where for convenience $a_i = m_A(x_i)$ and similarly for the b_i fit values. We will assume $p = 1$ but the following characterization of b_i^* is valid for any $p > 1$.

By orthogonality we know that a_i is at least as big as b_i^*. So first suppose $a_i = b_i^*$. This occurs if and only if no violation occurs: $a_i \leqq b_i$. (If this holds for all i, then $A = B^*$.) So $\max(0, a_i - b_i) = 0$. Next suppose $a_i > b_i^*$. This occurs if and only if a violation occurs: $a_i > b_i$. (If this holds for all i, then $B = B^*$.) So $b_i^* = b_i$ since B^* is the subset of B nearest to A. Equivalently, $a_i > b_i$ holds if and only if $\max(0, a_i - b_i) = a_i - b_i$. So the two cases together prove that $\max(0, a_i - b_i) = |a_i - b_i^*|$. Summing over all x_i gives:

$$d(A, B^*) = \sum_{i=1}^{n} \max(0, m_A(x_i) - m_B(x_i)).$$

So the two subsethood measures are equivalent.

This proof also proves a deeper characterization of the optimal subset B^*: $B^* = A \cap B$. For if a violation occurs, $a_i > b_i$ and $b_i = b_i^*$. So $\min(a_i, b_i) = b_i^*$. Otherwise $a_i = b_i^*$, and so $\min(a_i, b_i) = b_i^*$.

This in turn proves that B^* is a point of double optimality. Not only is B^* the subset of B nearest A, B^* is also A^*, the subset of A nearest to B: $d(B, F(2^A)) = d(B, A^*) = d(B, B^*)$. Figure 10 illustrates that $B^* = A \cap B = A^*$ is the set within both

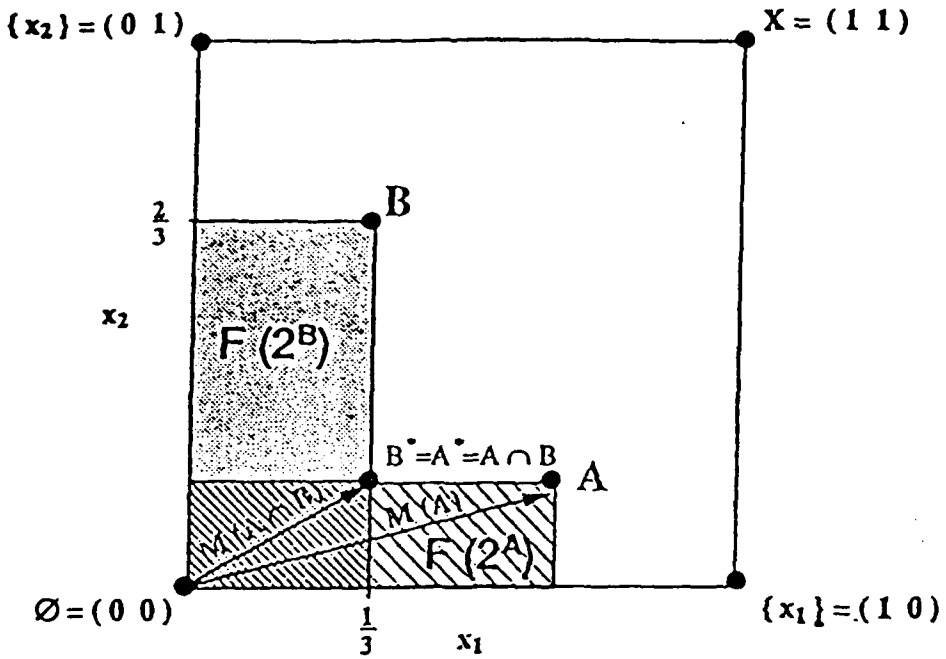


Figure 10 B^* as both the subset of B nearest A and the subset A^* of A nearest B: $B^* = A^* = A \cap B$. The distance $d(A, B^*) = M(A) - M(A \cap B)$ illustrates the Subsethood Theorem.

the hyper-rectangle $F(2^A)$ and the hyper-rectangle $F(2^B)$ that has maximal count $M(A \cap B)$.

Figure 10 also shows that the distance $d(A, B^*)$ is a vector magnitude difference: $d(A, B^*) = M(A) - M(A \cap B)$. Dividing both sides of this equality by $M(A)$ and rearranging proves a surprising and still deeper structural characterization of subsethood, the Subsethood Theorem.

SUBSETHOOD THEOREM

$$S(A, B) = \frac{M(A \cap B)}{M(A)}.$$

The ratio form of the subsethood measure $S(A, B)$ is familiar. It is the same as the ratio form of the conditional probability $P(B|A)$. The fundamental difference is that the ratio form is *derived* for the subsethood measure $S(A, B)$ but *assumed* for the conditional probability $P(B|A)$. This is the difference between showing and telling. The inability to derive conditional probability further suggests that probability is not real. For every probability is a conditional probability, $P(A) = P(A|X)$.

Consider first the physical interpretation of randomness as a relative frequency. The Subsethood Theorem suggests that randomness is a working fiction akin to the luminiferous ether of nineteenth-century physics—the phlogiston of thought. For in one stroke we can now derive the relative frequency definition of probability as $S(X, A)$, the degree to which a bivalent superset X, the sample space, is a subset of its own subset A. The concept of randomness never enters the deterministic framework.

Suppose A and B are nonfuzzy subsets of X. (X, like every observed set, is at

most countably infinite.) Suppose A is a subset of B. In the extreme case $B=X$. Then the degree of subsethood $S(B,A)$ is what has traditionally been called a *relative frequency*:

$$S(B,A)=\frac{M(A)}{M(B)}$$

$$=\frac{n_A}{N},$$

where the N elements of B constitute the *de facto* universe of discourse of the "experiment". (Of course the limit of the ratio $S(B,A)$ can be taken if it mathematically makes sense.[7]) The probability n_A/N has been reduced to degrees of subsethood, a purely fuzzy set-theoretical relationship. An immediate historical speculation is that if set theory had been more carefully worked out first, the notion of "randomness" might never have culturally evolved.

A classical example of relative frequency is the number n_A of successful trials in N trials. A biological example is the number of blue-eyed genes or alleles at all such chromosomal loci in a gene pool. The new way of expressing these relative frequencies $S(B,A)$ is the degree to which all trials are successes or all genes at a specific chromosomal location are for blue-eyedness. If the distinction between successful and unsuccessful trials is not clear cut, the resulting fuzzy relative frequency $S(B,A)$ may be real-valued. The frequency structure remains since A is a subset of B (since $B=X$ invariably in practice).

Where did the "randomness" go? The relative frequency $S(B,A)$ describes a fuzzy state of affairs, the degree to which B belongs to the power set of A: $S(B,A)=m_{2^A}(B)$. (Consider $B=X$ and $A=\{x_2\}$ in the unit square: the frequency $S(X,A)$ corresponds by the Pythagorean Theorem to the ratio of the left cube edge and the long diagonal to X.) Whether $S(B,A)$ is a rational or irrational number seems a technicality, a matter of fineness of quantization, if it is not zero or one. In practice only physical objects like tossed coins and DNA strands are involved. Their individual behavior might be fully determined by a system of differential equations.

The key quantity is the measure of overlap $M(A\cap B)$. This count does not involve "randomness". It counts which elements are identical or similar and to what degree. The phenomena themselves are deterministic. The corresponding frequency number that summarizes the deterministic situation is also deterministic. The same situation always gives the same number. The number may be used also to place bets or to switch a phone line, but it remains part of the description of a specific state of affairs. The deterministic subsethood derivation of relative frequency eliminates the need to invoke an undefined "randomness" to further describe the situation.

The identification of relative frequency with probability is cultural, not logical. This may take getting used to after hundreds of years of casting gambling intuitions as matters of probability and a century of building probability into the description of the universe. It is ironic that to date every assumption of probability—at least in the relative frequency sense of science, engineering, gambling, and daily life—has actually been an invocation of fuzziness.

9. BAYESIAN POLEMICS

Bayesian probabilists interpret probability as a subjective state of knowledge. In practice they use relative frequencies (subsethood degrees) but only to approximate these "states of knowledge".

Bayesianism is a polemical doctrine. Bayesians claim that they, and only they, use all and only the available uncertainty information in the description of uncertain phenomena. This stems from the Bayes Theorem expansion of the "*a posteriori*" conditional probability $P(H_i|E)$, the probability that H_i, the ith of k-many disjoint hypotheses $\{H_j\}$, is true when evidence E is observed:

$$P(H_i|E) = \frac{P(E \cap H_i)}{P(E)}$$

$$= \frac{P(E|H_i)P(H_i)}{P(E)}$$

$$= \frac{P(E|H_i)P(H_i)}{\sum_{j=1}^{k} P(E|H_j)P(H_j)},$$

since the hypotheses partition the sample space X: $H_1 \cup H_2 \cup \cdots \cup H_k = X$ and $H_i \cap H_j = \emptyset$ if $i \neq j$.

Conceptually, Bayesians use all available information in computing this posterior distribution by using the "*a priori*" or prior distribution $P(H_i)$ of the hypotheses. Mathematically, the Bayesian approach clearly stems from the ratio form of the conditional probability.

The Subsethood Theorem trivially implies Bayes Theorem when the hypotheses $\{H_i\}$ and evidence E are nonfuzzy subsets. More important, the Subsethood Theorem implies the Fuzzy Bayes Theorem in the more interesting case when the observed data E is fuzzy:

$$S(E, H_i) = \frac{S(H_i, E)M(H_i)}{\sum_{j=1}^{k} S(H_j, E)M(H_j)}$$

$$= \frac{S(H_i, E)f_i}{\sum_{j=1}^{k} S(H_j, E)f_j},$$

where

$$f_i = \frac{M(H_i)}{M(X)} = \frac{M(H_i)}{n} = S(X, H_i)$$

is the "relative frequency" of H_i, the degree to which all the hypotheses are H_i. So the Subsethood Theorem allows fuzzyists to be "Bayesians" as well.

The Subsethood Theorem implies inequality when the partitioning hypotheses are fuzzy. For instance, if $k = 2$, H^c is the complement of an arbitrary fuzzy set H, and evidence E is fuzzy, then[10] the occurrence of nondegenerate hypothesis overlap and underlap gives a lower bound on the posterior subsethood:

$$S(E,H) \geq \frac{S(H,E)f_H}{S(H,E)f_H + S(H^c,E)f_{H^c}},$$

where $f_H = S(X,H)$. The lower bound is an increasing function of $M(H)$, a decreasing function of $M(H^c)$. Since a like lower bound holds for $S(E,H^c)$, adding the two posterior subsethoods gives the additive inequality:

$$S(E,H) + S(E,H^c) \geq 1,$$

an inequality arrived at independently by Zadeh[17] by directly defining a "relative sigma-count" as the subsethood measure given by the Subsethood Theorem. If H is nonfuzzy, equality holds as in the additive law of conditional probability:

$$P(H|E) + P(H^c|E) = 1.$$

The Subsethood Theorem implies a deeper Bayes Theorem for arbitrary fuzzy sets, the Odds-Form Fuzzy Bayes Theorem:

$$\frac{S(A_1 \cap H, A_2)}{S(A_1 \cap H, A_2^c)} = \frac{S(A_2 \cap H, A_1)}{S(A_2^c \cap H, A_1)} \frac{S(H, A_2)}{S(H, A_2^c)}.$$

This theorem is proved directly by replacing the subsethood terms on the righthand side with their equivalent ratios of counts, canceling like terms three times, multiplying by $M(A_1 \cap H)/M(A_1 \cap H)$, rearranging, and applying the Subsethood Theorem a second time.

We have now developed enough fuzzy theory to examine critically the recent anti-fuzzy polemics of Lindley[13] and Jaynes[6] (and thus Cheeseman[2] who uses Jaynes' arguments). To begin we observe four more corollaries of the Subsethood Theorem:

i) $$0 \leq S(H,A) \leq 1,$$

ii) $$S(H,A) = 1 \quad \text{if } H \subset A,$$

iii) $$S(H, A_1 \cup A_2) = S(H,A_1) + S(H,A_2) - S(H, A_1 \cap A_2),$$

iv) $$S(H, A_1 \cap A_2) = S(H,A_1) S(A_1 \cap H, A_2).$$

Each relationship follows from the ratio form of $S(A,B)$. The third relationship uses the additivity of the count $M(A)$, which follows from $\min(x,y) + \max(x,y) = x + y$.

Now make the notational identification $S(H,A) = P(A|H)$. We then obtain the defining relationships of conditional probability proposed by Lindley:[13]

Convexity: $0 \leq P(A|H) \leq 1$ and $P(A|H) = 1$ if H implies A,

Addition: $P(A_1 \cup A_2|H) = P(A_1|H) + P(A_2|H) - P(A_1 \cap A_2|H)$,

Multiplication: $P(A_1 \cap A_2 | H) = P(A_1 | H) P(A_2 | A_1 \cap H)$.

"From these three rules", Lindley[13] tells us,

> all of the many, rich and wonderful results of the probability calculus follow. They may be described as the axioms of probability.

Lindley takes these as "unassailable" axioms:

> We really have no choice about the rules governing our measurement of uncertainty: they are dictated to us by the inexorable laws of logic.

Lindley proceeds to build a "coherence" argument around the Odds-Form Bayes Theorem, which he correctly deduces from the axioms as the equality:

$$\frac{P(A_2 | A_1 \cap H)}{P(A_2^c | A_1 \cap H)} = \frac{P(A_1 | A_2 \cap H)}{P(A_1 | A_2^c \cap H)} \frac{P(A_2 | H)}{P(A_2^c | H)},$$

where here we interpret A^c as not-A. "Any other procedure", we are told, "is incoherent." This polemic evaporates in the face of the above four subsethood corollaries and the Odds-Form Fuzzy Bayes Theorem. Ironically, rather than establish the primacy of axiomatic probability, Lindley seems to argue that it is fuzziness in disguise.

Another source of Bayesian probability polemic[2] is maximum entropy estimation. Here the axiomatic argument rests on the so-called Cox's Theorem.[3] Cox's Theorem is best presented by its most vocal proponent, physicist E. T. Jaynes.

According to Jaynes:[6]

> Cox proved that any method of inference in which we represent degrees of plausibility by real numbers, is necessarily either equivalent to Laplace's, or inconsistent,

where Laplace is cited as an early Bayesian probabilist. In fact Cox used *bivalent* logic (Boolean algebra) and other assumptions to show that, again according to Jaynes, the "conditions of consistency can be stated in the form of functional equations," namely the probabilistic product and sum rules:

$$P(A \cap B | C) = P(A | B \cap C) P(B | C),$$

$$P(B | A) + P(B^c | A) = 1.$$

The Subsethood Theorem implies

$$S(C, A \cap B) = S(B \cap C, A) S(C, B),$$

$$S(A, B) + S(A, B^c) \geqq 1,$$

with, as we have seen, equality holding for the second subsethood relationship when B is nonfuzzy, which is the case in the Cox–Jaynes setting.

In the probabilistic case overlap and underlap are degenerate. So

$$P(A \cap A^c | B) = P(\emptyset | B) = \frac{P(\emptyset)}{P(B)} = 0,$$

and $P(B | A \cap A^c) = P(B | \emptyset)$ is undefined. Yet in general $S(B, A \cap A^c) > 0$ and $S(A \cap A^c, B)$ is defined when A is fuzzy and B is fuzzy or nonfuzzy.

Jaynes' claim is either false or concedes that probability is a special case of fuzziness. For strictly speaking, since the subsethood measure $S(A, B)$ satisfies the multiplicative and additive laws specified by Cox and yet differs from the conditional probability $P(B|A)$, Jaynes' claim is false.

Presumably Jaynes was unaware of fuzzy sets. He seems to suggest that the only alternative uncertainty theory is the frequency theory of probability, a theory we have seen reduced to the subsethood measure $S(X, A)$. So if we restrict consideration to nonfuzzy sets A and B, equality holds in the above subsethood relations and Jaynes is right: probability and fuzziness coincide. But fuzziness exists, indeed abounds, outside this restriction and classical probability theory does not. So fuzzy theory is an extension of probability theory. Equivalently, probability then is a special case of fuzziness.

Incidentally, when one examines Cox's actual arguments,[3] one finds that Cox assumes that the uncertainty combination operators in question are continuously *twice differentiable*! Min and max are not twice differentiable. Technically, Cox's theorem does not apply.

10. THE ENTROPY–SUBSETHOOD THEOREM

The Fuzzy Entropy Theorem and the Subsethood Theorem were independently derived from first principles, from sets-as-points unit-cube geometry. Both theorems involve ratios of cardinalities. A connection is inevitable.

The Entropy–Subsethood Theorem shows that the connection occurs in terms of overlap $A \cap A^c$ and underlap $A \cup A^c$ (what else?). The theorem says fuzzy entropy can be eliminated in favor of subsethood. So subsethood emerges as the fundamental, characterizing quantity of fuzziness—and, arguably, of probability as well.

ENTROPY–SUBSETHOOD THEOREM

$$E(A) = S(A \cup A^c, A \cap A^c).$$

The theorem is proved by replacing B and A in the Subsethood Theorem respectively with overlap $A \cap A^c$ and underlap $A \cup A^c$. Since overlap is a (dominated-membership function) subset of underlap, the intersection of the two sets is just overlap.

The Entropy–Subsethood Theorem is a peculiar relationship. It says that fuzziness is the degree to which the superset $A \cup A^c$ is a subset of its own subset $A \cap A^c$, the extent to which the whole is a part of one of its own parts, a relationship forbidden by Western logic.

This relationship violates our ingrained Venn-diagram intuitions of unambiguous set inclusion. Only the midpoint of I^n yields total containment of underlap in overlap. The cube vertices yield no containment. This parallels in the extreme the relative frequency relationship $S(X, A) = n_A/N$, where a nonfuzzy superset X is to some degree a subset of one of its nonfuzzy subsets A.

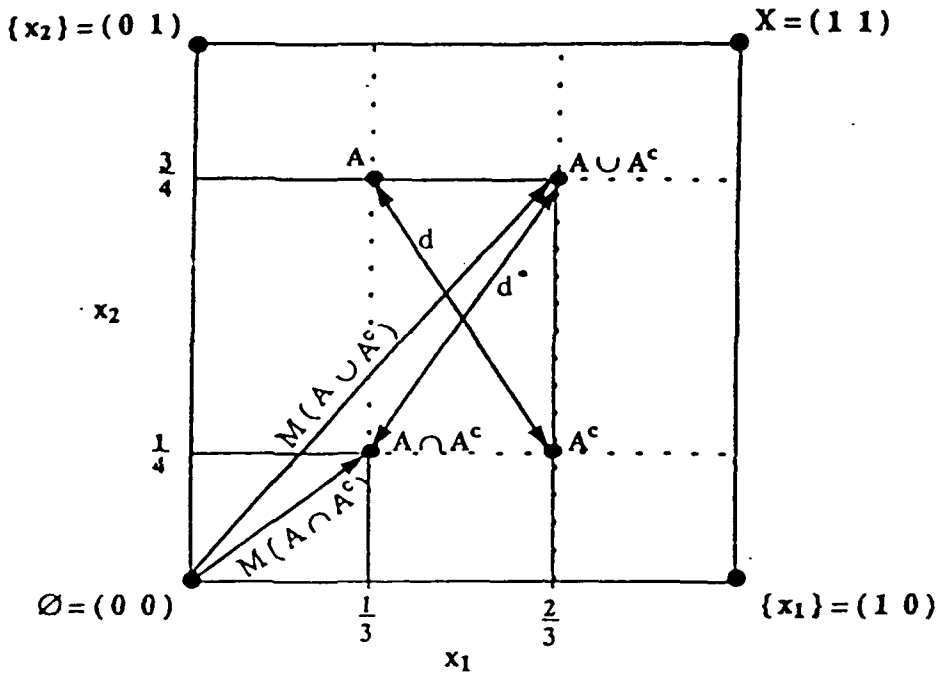


Figure 11 Entropy–Subsethood Theorem in two dimensions. Just as the long diagonals have equal length, $d(A, A^c) = d(A \cup A^c, A \cap A^c) = d^* = M(A \cup A^c) - M(A \cap A^c)$, the shortest distance from $A \cup A^c$ to the fuzzy power set of $A \cap A^c$.

Figure 11 illustrates the Entropy–Subsethood Theorem. It shows that d^*, the shortest distance from underlap $A \cup A^c$ to the hyper-rectangle defining the fuzzy power set of overlap $A \cap A^c$, is equivalent to $d(A \cup A^c, A \cap A^c) = d(A, A^c)$ and to a difference of vector magnitudes: $d^* = M(A \cup A^c) - M(A \cap A^c)$.

The Entropy–Subsethood Theorem implies that no probability measure measures fuzziness. For the moment, suppose not. Suppose fuzzy entropy measures nothing new; fuzziness is simply disguised probability. Suppose, as Lindley[13] claims, that probability theory "is adequate for all problems involving uncertainty." So there exists some probability measure P such that $P = E$. P cannot be identically zero because $P(X) = 1$. Then there is some A such that $P(A) = E(A) > 0$. But in a probability space there is no overlap or underlap: $A \cap A^c = \emptyset$ and $A \cup A^c = X$.

The Entropy–Subsethood Theorem then implies that $0 < P(A) = E(A) = S(A \cup A^c, A \cap A^c) = S(X, \emptyset)$. The only way X can be a subset to any degree of the empty set is if X itself, and hence A, is empty: $X = A = \emptyset$. Then the sure event X is impossible: $P(X) = P(\emptyset) = 0$. Or the impossible event is sure: $P(\emptyset) = 1$. Either outcome is a bivalent contradiction, impervious to normalization. So there exists no probability measure P that measures fuzziness. Fuzziness exists.

This *within-cube* theory can be extended[11] to define a natural fuzzy integral with respect to the fuzzy counting measure M. A more practical extension[11] is to mappings *between* fuzzy cubes, in particular to *fuzzy associative memories*. In short, a fuzzy set is a point in a unit hypercube I^n. A *fuzzy system* $S: I^n \to I^p$ is a mapping between cubes. Fuzzy systems map fuzzy subsets of the input space X to fuzzy subsets of the output space Y. Fuzzy systems are tools of machine intelligence, and can be applied to a wide range of control and decision problems.

11. PRECISE PAST, FUZZY FUTURE[4]

The boat of uncertainty reasoning is being rebuilt at sea. Plank by plank fuzzy theory is beginning to gradually shape its design. Today only a few fuzzy planks have been laid. But a hundred years from now, a thousand years from now, the boat of uncertainty reasoning may little resemble the boat of today. Notions and measures of overlap $A \cap A^c$ and underlap $A \cup A^c$ will have smoothed its rudder. Amassed fuzzy applications, hardware, and products will have broadened its sails. And no one on the boat will believe that there was a time when a concept as simple, as intuitive, as expressive[18] as a *fuzzy set* met with such impassioned denial.

How would the world be different today if fuzziness had been developed, taught, and applied before probability theory? Suppose the fuzzy framework was worked out at the time of Galileo of Laplace. Suppose Isaac Newton included an appendix on the geometry of fuzzy sets in his *Principia*. What would be different today?

Reasoning systems in machine intelligence would surely be different. So would be the range of automatic control devices. There would be many more of them, and they would more accurately reflect our reasoning processes than do our current decision trees and thermostats. Western belief systems might be more Eastern, and vice versa. (How many Westerners can name five Eastern books?) More of social science might be systematized. Historical tendencies would have been easier to articulate and defend. Communication, signal processing, and computational hardware might be built around the *fit*. Our physical explorations of subatomic reality, antimatter, and the spacetime fabric may have led to different times and places. Relative frequencies might be considered the everyday application of fuzzy subsethood. Besides betting on games of chance or frequency, betting on games of degree—perhaps involving simulated chaotic trajectories in unit cubes (or guppies swimming in hand-held cubical aquaria) or real-valued dice—might help support the economy of Las Vegas.

As the total amount of information in society continues to grow exponentially, the velocity of scientific and cultural change increases. Cultural change that once took centuries can now occur in a few years, perhaps soon in a single year. A current engineering example of this velocity of change is Moore's Law, the doubling of silicon-chip transistor density every one to two years.

One tendency of this information acceleration is to leave further behind what has already been explored. The complementary tendency is to soon experiment with systems that may at present seem distant, impractical, even absurd. In this light the recent developments in fuzzy theory and in fuzzy applications and hardware will surely affect the science, engineering, and culture of the future. The question is to what degree.

ACKNOWLEDGMENT

This research was supported by the Air Force Office for Scientific Research (AFOSR-88-0236) and by a grant from the Rockwell Science Center.

REFERENCES

1. W. Bandler and L. Kohout, "Fuzzy power sets and fuzzy implication operators." *Fuzzy Sets and Systems*, 4, 1980, pp. 13–30.

2. P. Cheeseman, "In defense of probability." *Proc. of the IJCAI-85*, Aug. 1985, pp. 1002–1009.
3. R. T. Cox, "Probability, frequency, and reasonable expectations." *American Journal of Physics*, **14**, No. 1 Jan./Feb. 1946, pp. 1–13.
4. B. R. Gaines, "Precise past, fuzzy future." *International Journal of Man–Machine Studies*, **19**, 1983, pp. 117–134.
5. D. Hume, *An Inquiry Concerning Human Understanding*, 1748.
6. E. T. Jaynes, "Where do we stand on maximum entropy?" In: *The Maximum Entropy Formalism*, edited by Levine and Tribus, MIT Press, Cambridge, Mass., 1979.
7. M. Kac, *Probability and Related Topics in Physical Sciences, Lectures in Applied Mathematics*, Vol. I. Interscience, New York, 1959.
8. G. J. Klir and T. A. Folger, *Fuzzy Sets, Uncertainty, and Information*. Prentice-Hall, Englewood Cliffs, New Jersey, 1988.
9. B. Kosko, "Counting with fuzzy sets." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8, July 1986, pp. 556–557.
10. B. Kosko, "Fuzzy entropy and conditioning." *Information Sciences*, **40**, 1986, pp. 165–174.
11. B. Kosko, *Foundations of Fuzzy Estimation Theory*. Ph.D. dissertation, Department of Electrical Engineering, University of California at Irvine, June 1987; Order Number 8801936, University Microfilms International, 300 N. Zeeb Road, Ann Arbor, Michigan 48106.
12. B. Kosko, "Fuzzy quantum states." in preparation, 1990.
13. D. V. Lindley, "The probability approach to the treatment of uncertainty in artificial intelligence and expert systems." *Statistical Science*, 2, No. 1, Feb. 1987, pp. 17–24.
14. H. Prade, "A computational approach to approximate and plausible reasoning with applications to expert systems." *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 7, 1985, pp. 260–283.
15. L. Wittgenstein, *Tractacus Logico-Philosophicus*. Routledge & Kegan Paul, London, 1922.
16. L. A. Zadeh, "Fuzzy sets." *Information and Control*, **8**, 1965, 338–353.
17. L. A. Zadeh, "A computational approach to fuzzy quantifiers in natural languages." *Computers and Mathematics*, **9**, No. 1, 1983, pp. 149–184.
18. L. A. Zadeh, "Making computers think like people." *IEEE Spectrum*, Aug. 1984, pp. 26–32.

Bart Kosko is an assistant professor in the electrical engineering department of the University of Southern California. He received bachelor degrees in philosophy and economics from USC, a masters degree in applied mathematics from UC San Diego, and a Ph.D. degree in electrical engineering from UC Irvine. Dr. Kosko is an elected member of the governing board of the International Neural Network Society, managing editor of Springer-Verlag's *Lecture Notes in Neural Computing* monograph series, and associate editor of *IEEE Transactions on Neural Networks*, *Journal of Mathematical Biology*, *Lecture Notes in Biomathematics*, and *Neural Networks*. Dr. Kosko was program chairman of the 1987 and 1988 IEEE International Conferences on Neural Networks and is program co-chairman of the 1990 International Joint Conference on Neural Networks and program chairman of the 1990 International Fuzzy Logic and Neural Network Conference in Iizuka, Japan. Dr. Kosko is a USC Shell Oil Faculty Fellow.

ADAPTIVE FUZZY SYSTEMS FOR BACKING UP A TRUCK-AND-TRAILER

Seong-Gon Kong and Bart Kosko
Department of Electrical Engineering
Signal and Image Processing Institute
University of Southern California
Los Angeles, California 90089-0272

Abstract

We developed fuzzy and neural-network control systems to back up a simulated truck, and truck-and-trailer, to a loading dock in a planar parking lot. The fuzzy systems performed well until we randomly removed over 50 % of their fuzzy-associative-memory (FAM) rules. They also performed well when we replaced key FAM equilibration rules with destructive or "sabotage" rules. We trained the neural network systems with the supervised backpropagatio.. learning algorithm and tested their robustness by removing random subsets of training data in learning sequences. The neural systems performed well but required extensive computation for training. We used unsupervised differential competitive learning (DCL), and product-space clustering, to adaptively generate FAM rules from training data. The original fuzzy and neural control systems generated trajectory data. The DCL system rapidly recovered the underlying FAM rules. Product-space clustering converted the neural truck systems into structured sets of FAM rules that approximated the neural system's behavior.

Fuzzy and Neural Control Systems

We construct fuzzy and neural control systems directly from control data, but from different types of control data. Fuzzy systems use a small number of structured *linguistic* input-output samples from an expert or from some other adaptive estimator. Neural systems use a large number of *numeric* input-output samples from the control process or from some other database. Adaptive fuzzy systems also use numeric control data.

Figure 1 illustrates this difference. The neural system estimates function $f : X \longrightarrow Y$ from several numerical *point* samples (x_i, y_i). The fuzzy system estimates f from a few fuzzy *set* samples or fuzzy associations (A_i, B_i).

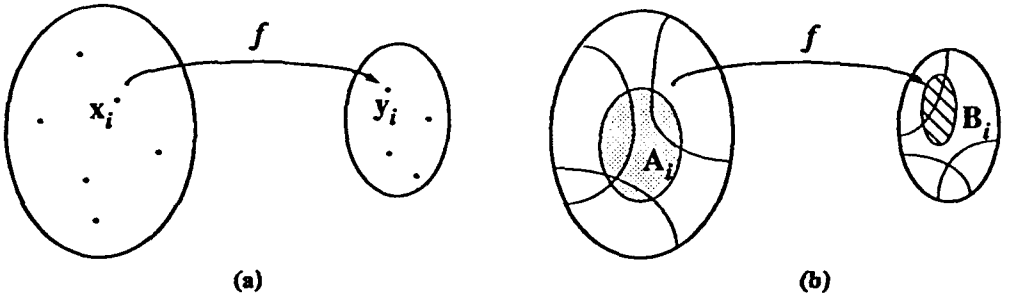


FIGURE 1 Geometry of neural and fuzzy function estimation. The neural approach (a) uses several numerical point samples. The fuzzy approach (b) uses a few fuzzy set samples.

Fuzzy and neural systems offer a key advantage over traditional control approaches. They offer *model-free estimation* of the control system. The user need not specify how the controller's output mathematically depends on its input. Instead the user provides a few common-sense associations of how the control variables behave. Or the user provides a statistically representative set of numerical training samples. Even if a math-model controller is available, fuzzy or neural controllers may prove more robust and easier to modify.

Which system, fuzzy or neural, performs better for which type of control problem de-

pends on the type and availability of sample data. If experts provide structured knowledge of the control process, or if sufficient numerical training samples are unavailable, the fuzzy approach may be preferable. We can construct a fuzzy control system with comparative ease when experts or fuzzy engineers provide accurate structured knowledge. A fuzzy control system seems a reasonable benchmark in such cases, even if we can develop a neural controller or math-model controller.

If we have representative numerical data but not structured expertise, the neural approach may be preferable. Or a statistical regression approach may be more appropriate. The data simply tell their own story—if there is a story to tell. Yet even here we can use a hybrid fuzzy-neural system, an adaptive fuzzy system. We can use the numerical data to generate *fuzzy associative memory* (FAM) rules. The FAM rules can then form the skeleton of a fuzzy control *architecture*. In short, if structured knowledge is unavailable, estimate it. This may be more practical than it would appear because of the small number of control FAM rules needed to reliably control many realworld processes.

How can we compare fuzzy and neural controllers? Abstract comparison proves difficult because both approaches build a control black box in different ways. That they build black boxes distinguishes them from math-model controllers. It also suggests we can compare them, at least approximately, by their black-box control performance.

Each control system generated an output *control surface* as it ranged over the common input space of parameter values. Figure 5 below shows three-dimensional control surfaces for the fuzzy and neural controllers. For control systems with few input parameters with moderately quantized ranges, we can store both fuzzy and neural controllers—or rather their quantized control surfaces—as decision look-up tables. Then once we specify a system performance criterion, we can in principle quantitatively compare the controllers.

Comparing system trajectories proved more complicated. In the case at hand, we wanted to back up a truck, and truck-and-trailer, to a loading dock. We can measure and compare the quality and quantity of the truck trajectory, perhaps with mean-squared error criteria. Intuitively, we preferred smooth short trajectories to jagged long trajectories. Reaching the loading-dock goal was also important. In practice it is the most important performance requirement. We must balance the trajectory type with the trajectory

destination, and this reduces to the pragmatic issue of balancing means and ends.

Below we develop a simple fuzzy control system and a simple neural control system for backing up a truck, and truck-and-trailer, in an open parking lot. The recent neural network truck backer-upper simulation of Nguyen and Widrow [1989] motivated our choice of control problem.

The fuzzy control system compared favorably with the neural controller in terms of black-box development effort, black-box computational load, smoothness of truck trajectories, and robustness.

We studied robustness of the fuzzy control systems in two ways. We deliberately added confusing FAM rules—"sabotage" rules—to the system, and we randomly removed different subsets of FAM rules. We studied robustness of the neural controller by randomly removing different portions of the training data in learning sequences. We also converted the neural control systems to structured FAM-bank systems.

Backing up a truck

Figure 2 shows the simulated truck and loading zone. The truck corresponds to the cab part of the neural truck in the Nguyen-Widrow neural truck backer-upper system. The three state variables ϕ, x, and y exactly determine the truck position. ϕ specifies the angle of the truck with the horizontal. The coordinate pair (x, y) specifies the position of the rear center of the truck in the plane.

The goal was to make the truck arrive at the loading dock at a right angle ($\phi_f = 90°$) and to align the position (x, y) of the truck with the desired loading dock (x_f, y_f). We considered only backing up. The truck moved backward by some fixed distance at every stage. The loading zone corresponded to the plane $[0, 100] \times [0, 100]$, and (x_f, y_f) equaled $(50, 100)$.

At every stage the fuzzy and neural controllers should produce the steering angle θ that backs up the truck to the loading dock from any initial position and from any angle in the loading zone.

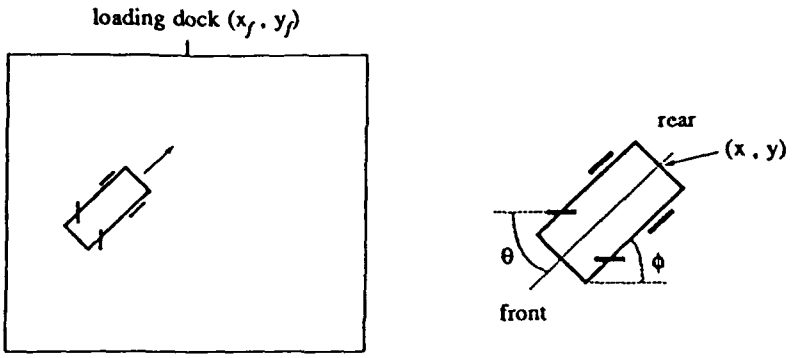


FIGURE 2 Diagram of simulated truck and loading zone.

Fuzzy Truck Backer-Upper System

We first specified each controller's input and output variables. The input variables were the truck angle ϕ and the x-position coordinate x. The output variable was the steering-angle signal θ. We assumed enough clearance between the truck and the loading dock so we could ignore the y-position coordinate. The variable ranges were as follows:

$$0 \leq x \leq 100 \ ,$$
$$-90 \leq \phi \leq 270 \ ,$$
$$-30 \leq \theta \leq 30 \ .$$

Positive values of θ represented clockwise rotations of the steering wheel. Negative values represented counterclockwise rotations. We discretized all values to reduce computation. The resolution of ϕ and θ was one degree each. The resolution of x was 0.1.

Next we specified the fuzzy-set values of the input and output fuzzy variables. The fuzzy sets numerically represented linguistic terms, the sort of linguistic terms an expert might use to describe the control system's behavior. We chose the fuzzy-set values of the fuzzy variables as follows:

Angle ϕ		x-position x		Steering-angle signal θ	
RB:	Right Below	LE:	Left	NB:	Negative Big
RU:	Right Upper	LC:	Left Center	NM:	Negative Medium
RV:	Right Vertical	CE:	Center	NS:	Negative Small
VE:	Vertical	RC:	Right Center	ZE:	Zero
LV:	Left Vertical	RI:	Right	PS:	Positive Small
LU:	Left Upper			PM:	Positive Medium
LB:	Left Below			PB:	Positive Big

Fuzzy subsets contain elements with degrees of membership. A fuzzy membership function $m_A : Z \longrightarrow [0,1]$ assigns a real number between 0 and 1 to every element z in the universe of discourse Z. This number $m_A(z)$ indicates the degree to which the object or data z belongs to the fuzzy set A. Equivalently, $m_A(z)$ defines the *fit* (fuzzy unit) value [Kosko, 1986] of element z in A.

Fuzzy membership functions can have different shapes depending on the designer's preference or experience. In practice fuzzy engineers have found triangular and trapezoidal shapes help capture the modeler's sense of fuzzy numbers and simplify computation. Figure 3 shows membership-function graphs of the fuzzy subsets above. In the third graph, for example, $\theta = 20°$ is Positive Medium to degree 0.5, but only Positive Big to degree 0.3.

In Figure 3 the fuzzy sets CE, VE, and ZE are narrower than the other fuzzy sets. These narrow fuzzy sets permit fine control near the loading dock. We used wider fuzzy sets to describe the endpoints of the range of the fuzzy variables ϕ, x, and θ. The wider fuzzy sets permitted rough control far from the loading dock.

Next we specified the fuzzy "rulebase" or bank of *fuzzy associative memory* (FAM) rules. Fuzzy associations or "rules" (A, B) associate output fuzzy sets B of control values with input fuzzy sets A of input-variable values. We can write fuzzy associations as antecedent-consequent pairs or IF-THEN statements.

In the truck backer-upper case, the FAM bank contained the 35 FAM rules in Figure 4. For example, the FAM rule of the left upper block (FAM rule 1) corresponds to the following

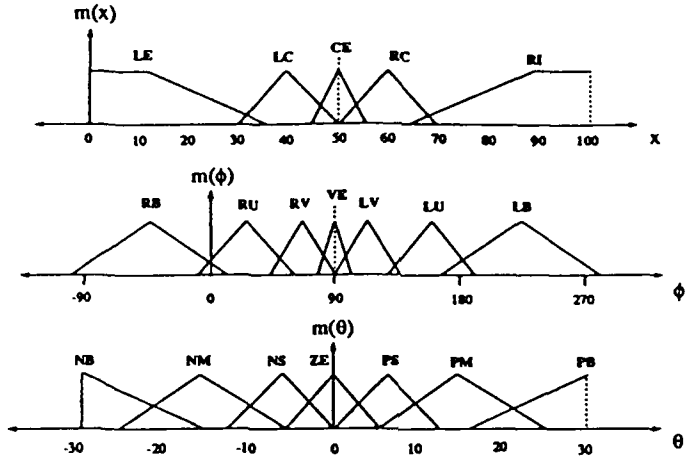


FIGURE 3 Fuzzy membership functions for each linguistic fuzzy-set value. To allow finer control, the fuzzy sets that correspond to near the loading dock are narrower than the fuzzy sets that correspond to far from the loading dock.

fuzzy association:

$$IF\ x = LE\ AND\ \phi = RB, \qquad THEN\ \theta = PS.$$

FAM rule 18 indicates that if the truck is in near the equilibrium position, then the controller should not produce a positive or negative steering-angle signal. The FAM rules in the FAM-bank matrix reflect the symmetry of the controlled system.

For the initial condition $x = 50$ and $\phi = 270$, the fuzzy truck did not perform well. The symmetry of the FAM rules and the fuzzy sets cancelled the fuzzy controller output in a rare saddle point. For this initial condition, the neural controller (and truck-and-trailer below) also performed poorly. Any perturbation breaks the symmetry. For example, the rule (If $x = 50$ and $\phi = 270$, then $\theta = 5$) corrected the problem.

The three-dimensional control surfaces in Figure 5 show steering-angle signal outputs θ that correspond to all combinations of values of the two input state variables ϕ and x. The control surface defines the fuzzy controller. In this simulation the correlation-minimum FAM inference procedure, discussed in [Kosko, 1990a], determined the fuzzy control surface. If the control surface changes with sampled variable values, the system

ϕ \ x	LE	LC	CE	RC	RI
RB	[1] PS	[2] PM	[3] PM	[4] PB	[5] PB
RU	[6] NS	[7] PS	PM	PB	PB
RV	NM	NS	PS	PM	PB
VE	NM	NM	[18] ZE	PM	PM
LV	NB	NM	NS	PS	PM
LU	NB	NB	NM	NS	PS
LB	NB	NB	NM	NM	[35] NS

FIGURE 4 FAM-bank matrix for the fuzzy truck backer-upper controller.

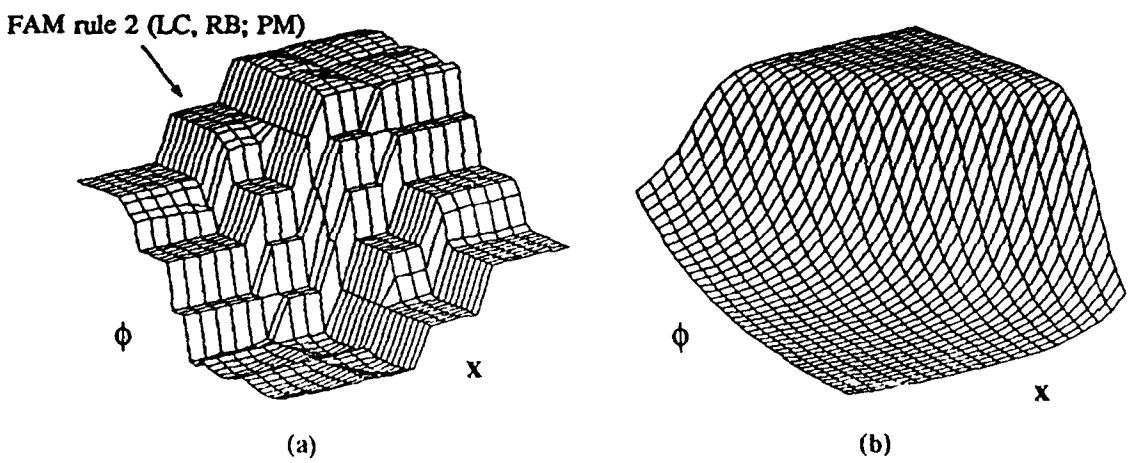


FIGURE 5 (a) Control surface of the fuzzy controller. Fuzzy-set values determined the input and output combination corresponding to FAM rule 2 (IF x=LC AND ϕ=RB, THEN θ=PM). (b) Corresponding control surface of the neural controller for constant value y=20.

behaves as an *adaptive* fuzzy controller. Below we demonstrate unsupervised adaptive control of the truck and the truck-and-trailer systems.

Finally, we determined the output action given the input conditions. We used the correlation-minimum inference method illustrated in Figure 6. Each FAM rule produced the output fuzzy set clipped at the degree of membership determined by the input conditions and the FAM rule. Alternatively, correlation-product inference [Kosko, 1990a] would combine FAM rules multiplicatively. Each FAM rule emitted a fit-weighted output fuzzy set O_i at each iteration. The total output O added these weighted outputs:

$$O = \sum_i O_i \tag{1}$$

$$= \sum_i \min(f_i, S_i) \quad , \tag{2}$$

where f_i denotes the antecedent fit value and S_i represents the consequent fuzzy set of steering-angle values in the ith FAM rule. Earlier fuzzy systems combined the output sets O_i with pairwise maxima. But this tends to produce a uniform output set O as the number of FAM rules increases. Adding the output sets O_i invokes the fuzzy version of the Central Limit Theorem. This tends to produce a symmetric, unimodal output fuzzy set O of steering-angle values.

Fuzzy systems map fuzzy sets to fuzzy sets. The fuzzy control system's output defines the fuzzy set O of steering-angle values at each iteration. We must "defuzzify" the fuzzy set O to produce a numerical (point-estimate) steering-angle output value θ.

As discussed in [Kosko, 1990a], the simplest defuzzification scheme selects the value corresponding to the *maximum fit* value in the fuzzy set. This mode-selection approach ignores most of the information in the output fuzzy set and requires an additional decision algorithm when multiple modes occur.

Centroid defuzzification provides a more effective procedure. This method uses the *fuzzy centroid* $\bar{\theta}$ as output:

$$\bar{\theta} = \frac{\sum_{j=1}^{p} \theta_j \, m_O(\theta_j)}{\sum_{j=1}^{p} m_O(\theta_j)} \quad , \tag{3}$$

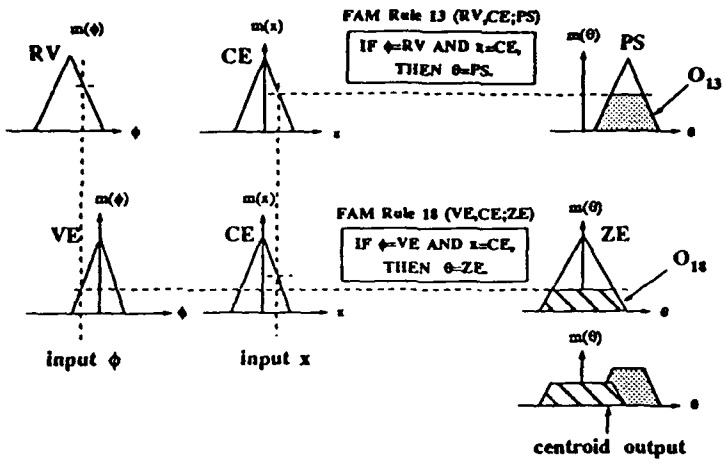


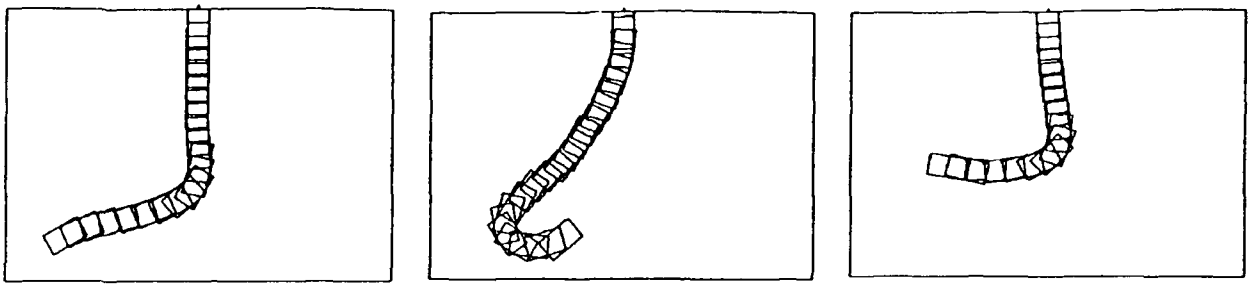
FIGURE 6 Correlation-minimum inference with centroid defuzzification method. Then FAM-rule antecedents combined with AND use the *minimum* fit value to activate consequents. Those combined with OR would use the *maximum* fit value.

where O defines a fuzzy subset of the steering-angle universe of discourse $\Theta = \{\theta_1, \ldots, \theta_p\}$. The central-limit-theorem effect produced by adding output fuzzy set O_i benefits both max-mode and centroid defuzzification. Figure 6 shows the correlation-minimum inference and centroid defuzzification applied to FAM rules 13 and 18. We used centroid defuzzification in all simulations.

With 35 FAM rules, the fuzzy truck controller produced successful truck backing-up trajectories starting from any initial position. Figure 7 shows typical examples of the fuzzy-controlled truck trajectories from different initial positions. The fuzzy control system did not use ("fire") all FAM rules at each iteration. Equivalently most output consequent sets are empty. In most cases the system used only one or two FAM rules at each iteration. The system used at most 4 FAM rules at once.

Neural Truck Backer-Upper System

The neural truck backer-upper of Nguyen and Widrow [1989] consisted of multilayer



(a) (b) (c)

FIGURE 7 Sample truck trajectories of the fuzzy controller for initial positions (x, y, ϕ): (a) (20,20,30), (b) (30,10,220), and (c) (30,40,−10).

feedforward neural networks trained with the backpropagation gradient-descent (stochastic-approximation) algorithm. The *neural control system* consisted of two neural networks: the controller network and the truck emulator network. The *controller network* produced an appropriate steering-angle signal output given any parking-lot coordinates (x, y), and the angle ϕ. The *emulator network* computed the next position of the truck. The emulator network took as input the previous truck position and the current steering-angle output computed by the controller network.

We did not train the emulator network since we could not obtain "universal" synaptic connection weights for the truck emulator network. The backpropagation learning algorithm did not converge for some sets of training samples. The number of training samples for the emulator network might exceed 3000. For example, the combinations of training samples of a given angle ϕ, x-position, y-position, and steering angle signal θ might correspond to 3150 ($18 \times 5 \times 5 \times 7$) samples depending on the division of the input-output product space. Moreover, the training samples were numerically similar since the neuronal signals assumed scaled values in $[0, 1]$ or $[-1, 1]$. For example, we treated close values, such as 0.40 and 0.41, as distinct sample values.

Simple kinematic equations replaced the truck emulator network. If the truck moved

backward from (x, y) to (x', y') at an iteration, then

$$x' = x + r\cos(\phi') \ , \tag{4}$$

$$y' = y + r\sin(\phi') \ , \tag{5}$$

$$\phi' = \phi + \theta \ . \tag{6}$$

r denotes the fixed driving distance of the truck for all backing movements. We used equations (4)–(6) instead of the emulator network. This did not affect the post-training performance of the neural truck backer-upper since the truck emulator network back-propagated only errors.

We trained only the controller network with backpropagation. The controller network used 24 "hidden" neurons with logistic sigmoid functions. In the training of the truck-controller, we estimated the ideal steering-angle signal at each stage before we trained the controller network. In the simulation, we used the arc-shaped truck trajectory produced by the fuzzy controller as the ideal trajectory. The fuzzy controller generated each training sample (x, y, ϕ, θ) at each iteration of the backing-up process. We used 35 training sample vectors and needed more than 100,000 iterations to train the controller network.

Figure 5b shows the resulting neural control surface for $y = 20$. The neural control surface shows less structure than the corresponding fuzzy control surface. This reflects the unstructured nature of black-box supervised learning. Figure 8 shows the network connection topology for our neural truck backer-upper control system.

Figure 9 shows typical examples of the neural-controlled truck trajectories from several initial positions. Even though we trained the neural network to follow the smooth arc-shaped path, some learned truck trajectories were non-optimal.

Comparison of Fuzzy and Neural Systems

As shown in Figure 7 and 9, the fuzzy controller always smoothly backed up the truck but the neural controller did not. The neural-controlled truck sometimes followed an irregular path.

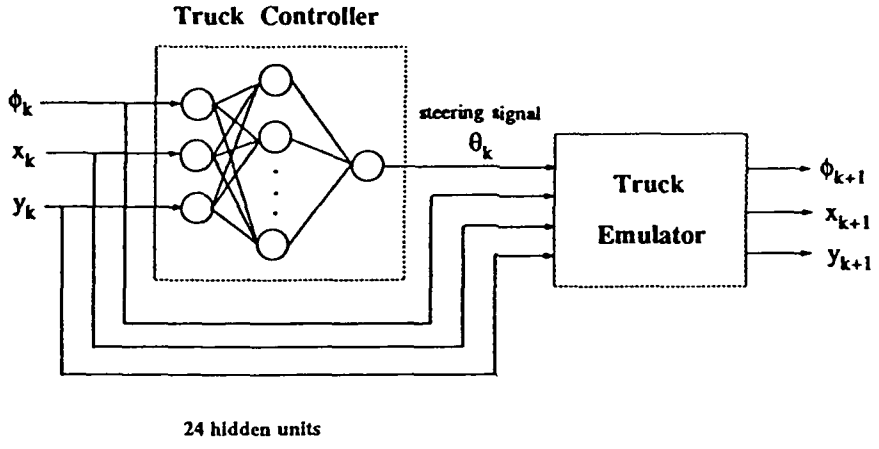


FIGURE 8 Topology of our neural control system.

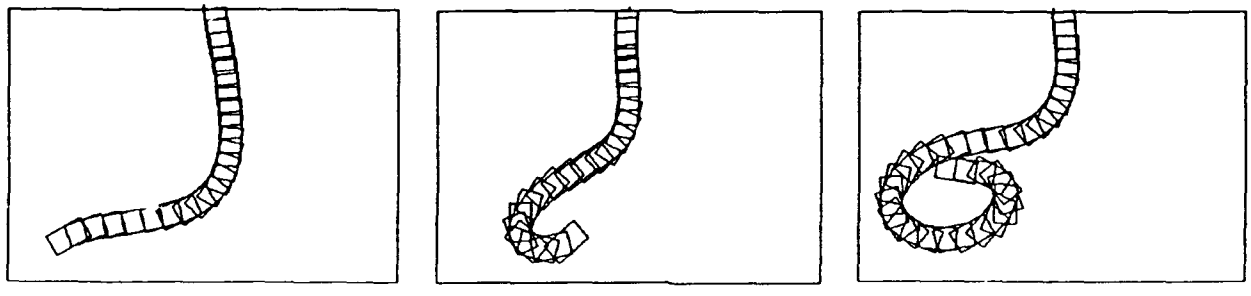
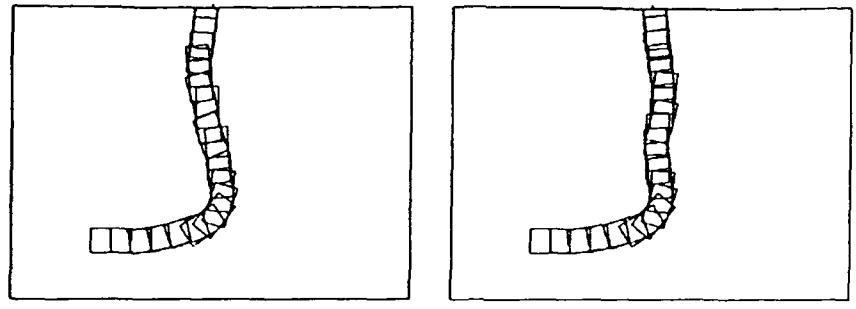


FIGURE 9 Sample truck trajectories of the neural controller for initial positions (x, y, ϕ): (a) (20,20,30), (b) (30,10,220), and (c) (30,40,-10).



(a) (b)

FIGURE 10 The fuzzy truck trajectory after we replaced the key steady-state FAM rule 18 by the two worst rules: (a) IF $x = CE$ AND $\phi = VE$, THEN $\theta = PB$, and (b) IF $x = CE$ AND $\phi = VE$, THEN $\theta = NB$.

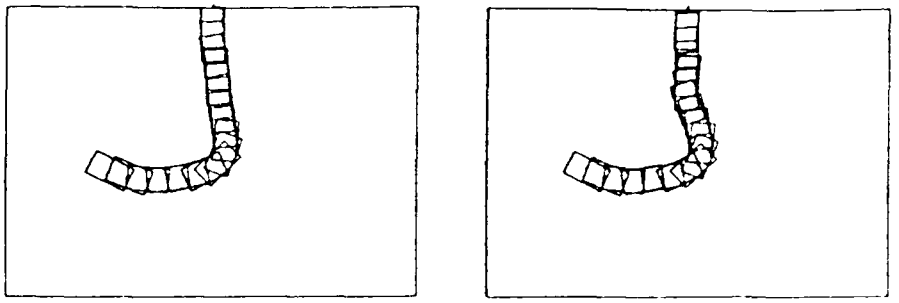
Training the neural control system was time-consuming. The backpropagation algorithm required thousands of back-ups to train the controller network. In some cases, the learning algorithm did not converge.

We "trained" the fuzzy controller by encoding our own common sense FAM rules. Once we develop the FAM-rule bank, we can compute control outputs from the resulting FAM-bank matrix or control surface. The fuzzy controller did not need a truck emulator and did not require a math model of how outputs depended on inputs.

The fuzzy controller was computationally lighter than the neural controller. Most computation operations in the neural controller involved the multiplication, addition, or logarithm of two real numbers. In the fuzzy controller, most computational operations involved comparing and adding two real numbers.

Sensitivity Analysis

We studied the sensitivity of the fuzzy controller in two ways. We replaced the FAM rules with destructive or "sabotage" FAM rules, and we randomly removed FAM rules.



(a) (b)

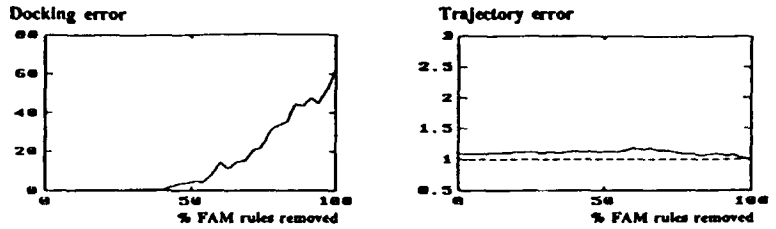
FIGURE 11 Fuzzy truck trajectory when (a) no FAM rules are removed and (b) FAM rules 7, 13, 18 and 23 are removed.

We deliberately chose sabotage FAM rules to confound the system. Figure 10 shows the trajectory when two sabotage FAM rules replaced the important steady-state FAM rule—FAM rule 18: the fuzzy controller should produce zero output when the truck is nearly in the correct parking position. Figure 11 shows the truck trajectory after we removed four randomly chosen FAM rules (7, 13, 18, and 23). These perturbations did not significantly affect the fuzzy controller's performance.

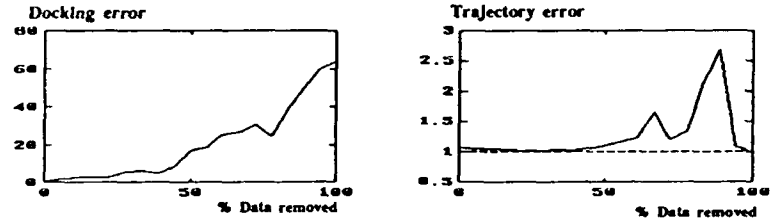
We studied robustness of each controller by examining failure rates. For the fuzzy controller we removed fixed percentages of randomly selected FAM rules from the system. For the neural controller we removed training data. Figure 12 shows performance errors averaged over ten typical back-ups with missing FAM rules for the fuzzy controller and missing training data for the neural controller. The missing FAM rules and training data ranged from 0 % to 100 % of the total. In Figure 12a, the docking error equaled the Euclidean distance from the actual final position (ϕ, x, y) to the desired final position (ϕ_f, x_f, y_f):

$$\text{Docking Error} = \sqrt{(\phi_f - \phi)^2 + (x_f - x)^2 + (y_f - y)^2} \quad . \tag{7}$$

In Figure 12b, the trajectory error equaled the ratio of the actual trajectory length of the



(a) Fuzzy controller



(b) BP-Neural controller

FIGURE 12 Comparison of robustness of the controllers: (a) Docking and Trajectory error of the fuzzy controller, (b) Docking and Trajectory error of the neural controller.

truck divided by the straight line distance to the loading dock:

$$\text{Trajectory Error} = \frac{\text{length of truck trajectory}}{\text{distance(initial position, desired final position)}}. \qquad (8)$$

Adaptive Fuzzy Truck Backer-Upper

Adaptive FAM (AFAM) systems generate FAM rules directly from training data. A one-dimensional FAM system, $S : I^n \longrightarrow I^p$, defines a FAM rule, a single association of the form (A_i, B_i). In this case the input-output product space equals $I^n \times I^p$. As discussed in [Kosko, 1990a], a FAM rule (A_i, B_i) defines a cluster or ball of points in the product-space cube $I^n \times I^p$ centered at the point (A_i, B_i). Adaptive clustering algorithms can estimate the

unknown FAM rule (A_i, B_i) from training samples in R^2. We used differential competitive learning (DCL) to recover the bank of FAM rules that generated the truck training data.

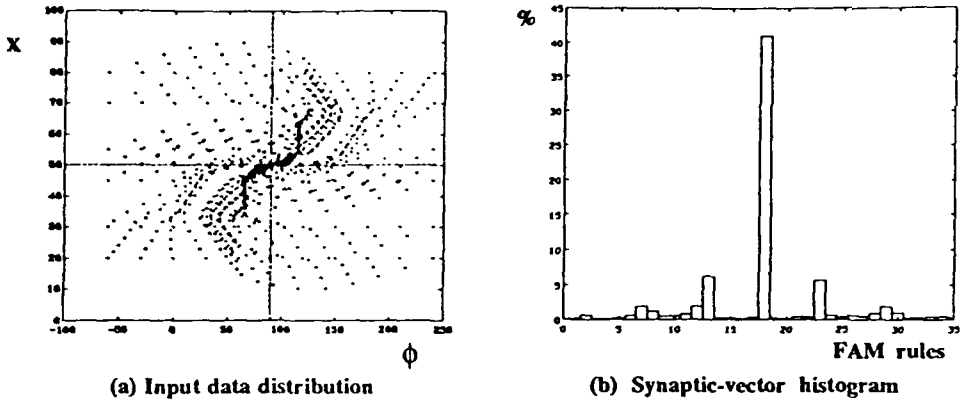
We generated 2230 truck samples from 7 different initial positions and varying angles. We chose the initial positions (20,20), (30,20), (45,20), (50,20), (55,20), (70,20), and (80,20). We changed the angle from $-60°$ to $240°$ at each initial position. At each step, the fuzzy controller produced output steering angle θ. The training vectors (x, ϕ, θ) defined points in a three-dimensional product-space. x had 5 fuzzy set values: LE, LC, CE, RC, and RI. ϕ had 7 fuzzy set values: RB, RU, RV, VE, LV, LU, and LB. θ had 7 fuzzy set values: NB, NM, NS, ZE, PS, PM, and PB. So there were 245 $(5 \times 7 \times 7)$ possible FAM cells.

We defined FAM cells by partitioning the effective product-space. FAM cells near the center were smaller than outer FAM cells because we chose narrow membership functions near the steady-state FAM cell. Uniform partitions of the product-space produced poor estimates of the original FAM rules. As in Figure 3, this reflected the need to judiciously define the fuzzy-set values of the system fuzzy variables.

We performed product-space clustering with the version of DCL discussed in [Kosko, 1990a]. If a FAM cell contained at least one of the 245 synaptic quantization vectors, we entered the corresponding FAM rule in the FAM matrix.

Figure 13a shows the input sample distribution of (x,ϕ). We did not include the variable θ in the figure. Training data clustered near the steady-state position ($x = 50$ and $\phi = 90°$). Figure 13b displays the synaptic-vector histogram after DCL classified 2230 training vectors for 35 FAM rules. Since successful FAM system generated the training samples, most training samples, and thus most synaptic vectors, clustered in the steady-state FAM cell.

DCL product-space clustering estimated 35 new FAM rules. Figure 14 shows the DCL-estimated FAM bank and the corresponding control surface. The DCL-estimated control surface visually resembles the underlying unknown control surface in Figure 5a. The two systems produce nearly equivalent truck-backing behavior. This suggests adaptive product-space clustering can estimate the FAM rules underlying expert behavior in many cases, even when the expert or fuzzy engineer cannot articulate the FAM rules.

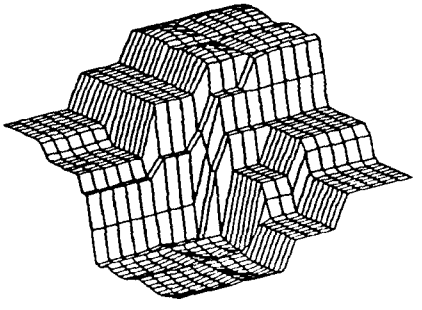


(a) Input data distribution (b) Synaptic-vector histogram

FIGURE 13 (a) Input data distribution, (b) Synaptic-vector histogram. Differential competitive learning allocated synaptic quantization vectors to FAM cells. The steady-state FAM cell (CE, VE; ZE) contained the most synaptic vectors.

ϕ \ x	LE	LC	CE	RC	RI
RB	PS	PM	PM	PB	PB
RU	ZE	PM	PM	PB	PB
RV	NS	ZE	PS	PM	PB
VE	NM	NM	ZE	PM	PM
LV	NB	NB	NS	PS	PS
LU	NB	NB	NB	NS	PS
LB	NB	NB	NM	NM	NS

(a)

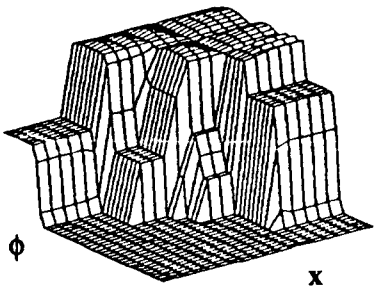


(b)

FIGURE 14 (a) DCL-estimated FAM bank. (b) Corresponding control surface.

φ \ x	LE	LC	CE	RC	RI
RB	PS	PB	PB	PB	PB
RU	NM	ZE	PM	PB	PB
RV	NM	NM	NS	PS	PB
VE	NM	NM	NM	ZE	PB
LV	NM	NM	NM	NS	PB
LU	NM	NM	NM	NM	PM
LB	NM	NM	NM	NM	NM

(a)



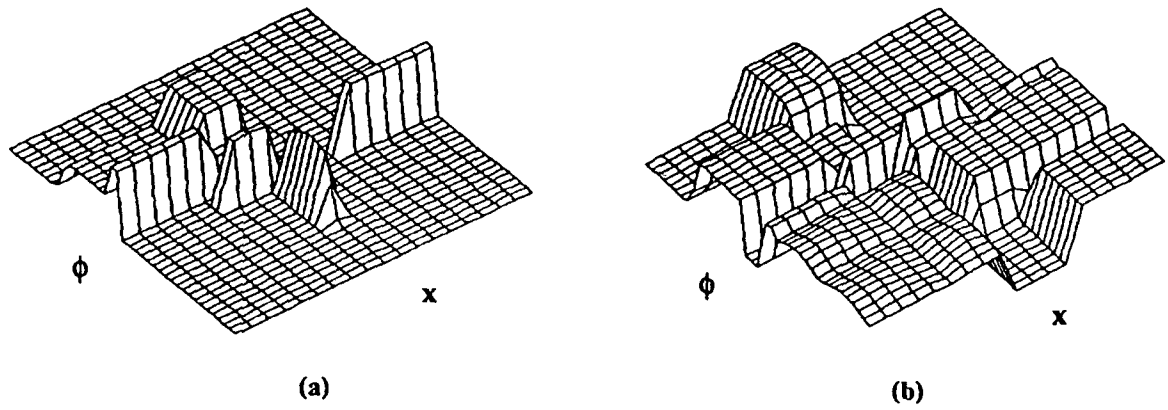
(b)

FIGURE 15 (a) FAM bank generated by the neural control surface in Figure 5b. (b) Control surface of the neural BP-AFAM system in (a).

We also used the neural control surface in Figure 5b to estimate FAM rules. We divided the input-output product-space into FAM cells as in the fuzzy control case. If the neural control surface intersected the FAM cell, we entered the corresponding FAM rule in a FAM bank. We averaged all neural control-surface values in a square region over the two input variables x and ϕ. We assigned the average value to one of 7 output fuzzy sets. Figure 15 shows the resulting FAM bank and corresponding control surface generated by the neural control surface in Figure 5b. This new control surface resembles the original fuzzy control surface in Figure 5a more than it resembles the neural control surface in Figure 5b. Note the absence of a steady-state FAM rule in the FAM matrix in Figure 5a.

Figure 16 compares the DCL-AFAM and BP-AFAM control surfaces with the fuzzy control surface in Figure 5a. Figure 16 shows the absolute difference of the control surfaces. As expected, the DCL-AFAM system produced less absolute error than the BP-AFAM system produced.

Figure 17 shows the docking and trajectory errors of the two AFAM control systems. The DCL-AFAM system produced less docking error than the BP-AFAM system produced for 100 arbitrary backing-up trials. The two AFAM systems generated similar backing-up trajectories. This suggests that black-box neural estimators can defir. : the front-end of



(a) (b)

FIGURE 16 (a) Absolute difference of the FAM surface in Figure 5a and the DCL-estimated FAM surface in Figure 14b. (b) Absolute difference of the FAM surface in Figure 5a and the neural-estimated FAM surface in Figure 15b.

FAM-structured systems. In principle we can use this technique to generate structured FAM rules for *any* neural application. We can then inspect and refine these rules and perhaps replace the original neural system with the tuned FAM system.

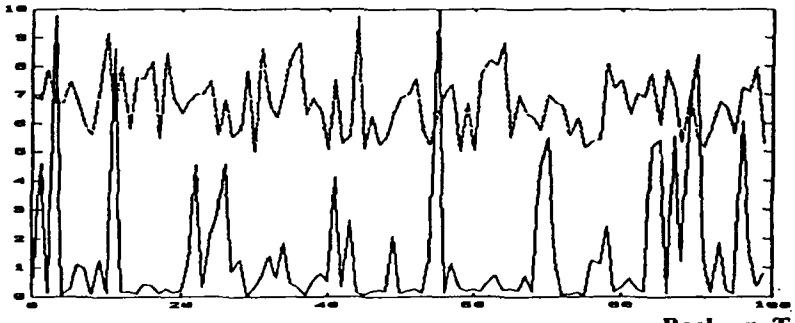
Fuzzy Truck-and-Trailer Controller

We added a trailer to the truck system, as in the original Nguyen-Widrow model. Figure 18 shows the simulated truck-and-trailer system. We added one more variable (cab angle, ϕ_c) to the three state variables of the trailerless truck. In this case a FAM rule takes the form

$$IF\ x = LE\ AND\ \phi_t = RB\ AND\ \phi_c = PO, \qquad THEN\ \beta = NS.$$

The four state variables x, y, ϕ_t, and ϕ_c determined the position of the truck-and-trailer system in the plane. Fuzzy variable ϕ_t corresponded to ϕ for the trailerless truck. Fuzzy variable ϕ_c specified the relative cab angle with respect to the center line along the trailer. ϕ_c ranged from $-90°$ to $90°$. The extreme cab angles $90°$ and $-90°$ corresponded to two

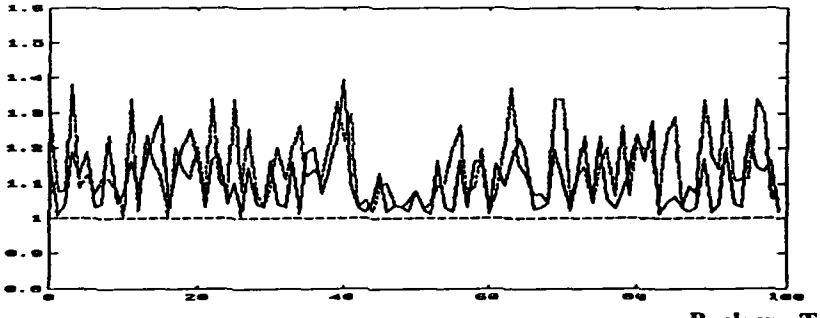
(a) Docking Error



Back-up Trial

- DCL-AFAM (solid) : mean = 1.4449, s.d. = 2.2594
- BP-AFAM (dashed) : mean = 6.6863, s.d. = 1.0665

(b) Trajectory Error



Back-up Trial

- DCL-AFAM (solid) : mean = 1.1075, s.d. = 0.0839
- BP-AFAM (dashed) : mean = 1.1453, s.d. = 0.1016

FIGURE 17 (a) Docking errors and (b) Trajectory errors of the DCL-AFAM and BP-AFAM control systems.

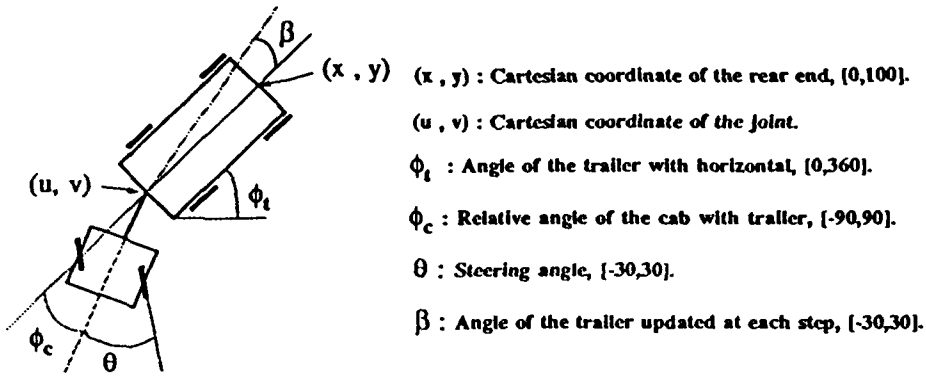


FIGURE 18 Diagram of the simulated truck-and-trailer system.

"jackknife" positions of the cab with respect to the trailer. Positive ϕ_c value indicated that the cab resided on the left-hand side of the trailer. Negative value indicated that it resided on the right-hand side. Figure 18 shows a positive angle value of ϕ_c.

Fuzzy variables x, ϕ_t, and ϕ_c defined the input variables. Fuzzy variable β defined the output variable. β measured the angle that we needed to update the trailer at each iteration. We computed the steering-angle output θ with the following geometric relationship. With the output β value computed, the trailer position (x, y) moved to the new position (x', y'):

$$x' = x + r\cos(\phi_t + \beta), \tag{9}$$

$$y' = y + r\sin(\phi_t + \beta), \tag{10}$$

where r denotes a fixed backing distance. Then the joint of the cab and the trailer (u, v) moved to the new position (u', v'):

$$u' = x' - \ell\cos(\phi_t + \beta), \tag{11}$$

$$v' = y' - \ell\sin(\phi_t + \beta), \tag{12}$$

where ℓ denotes the trailer length. We updated the directional vector $(dirU, dirV)$, which defined the cab angle, by

$$dirU' = dirU + \Delta u, \tag{13}$$

$$dirV' = dirV + \Delta v, \tag{14}$$

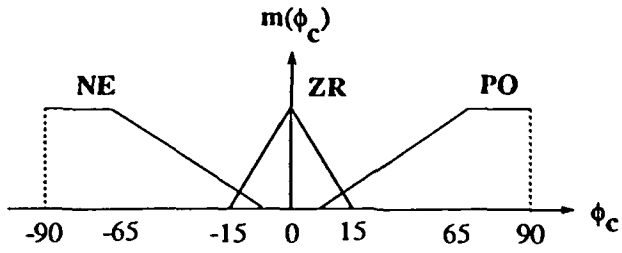


FIGURE 19 Membership graphs of the three fuzzy-set values of fuzzy variable ϕ_c.

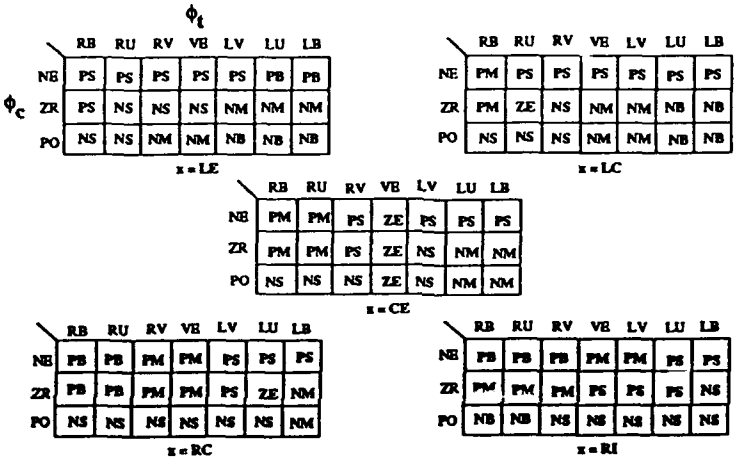
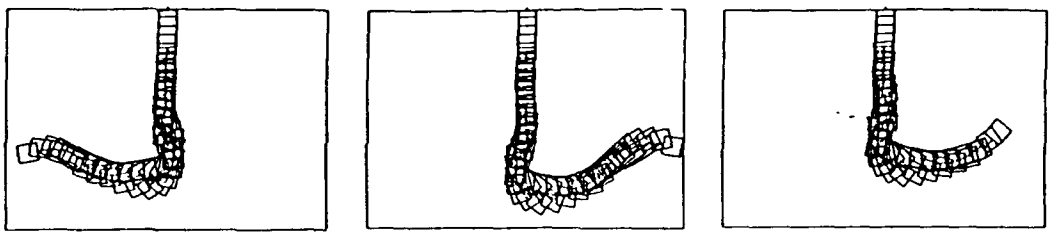


FIGURE 20 FAM bank of the fuzzy truck-and-trailer control system.

where $\Delta u = u' - u$, and $\Delta v = v' - v$. The new directional vector $(dirU', dirV')$ defines the new cab angle ϕ_c'. Then we obtain the steering angle value as $\theta = \phi_{c,h}' - \phi_{c,h}$, where $\phi_{c,h}$ denotes the cab angle with the horizontal. We chose the same fuzzy-set values and membership functions for β as we chose for θ. β ranged from $-30°$ to $30°$. We chose the fuzzy-set values of ϕ_c as NE, ZR and PO as in Figure 19.

Figure 20 displays the 5 FAM-rule matrices in the FAM bank of the fuzzy truck-and-trailer system. In Figure 20 we fixed the fuzzy variable x as LE, LC, CE, RC, and RI. There were 735 ($7 \times 5 \times 7 \times 3$) possible FAM rules and only 105 actual FAM rules.

Figure 21 shows typical backing-up trajectories of the fuzzy truck-and-trailer control



(a) (b) (c)

FIGURE 21 Sample truck-and-trailer trajectories from the fuzzy controller for initial positions (x, y, ϕ_t, ϕ_t): (a) (25, 30, −20, 30), (b) (80, 30, 210, −40), and (c) (70, 30, 200, 30).

system from different initial positions. The truck-and-trailer backed up in different directions depending on the relative position of the cab with respect to the trailer. The fuzzy control systems successfully controlled the truck-and-trailer in jackknife positions.

BP Truck-and-Trailer Control Systems

We added the cab-angle variable ϕ_c as to the backpropagation-trained neural truck controller as an input. The controller network contained 24 hidden neurons with output variable β. The training samples consisted of 5-dimensional space of the form $(x, y, \phi_t, \phi_c, \beta)$. We trained the controller network with 52 training samples from the fuzzy controller: 26 samples for the left half of the plane, 26 samples for the right half of the plane. We used equations (9)–(14) instead of the emulator network. Training required more than 200,000 iterations. Some training sequences did not converge. The BP-trained controller performed well except in a few cases. Figure 22 shows typical backing-up trajectories of the BP truck-and-trailer control system from the same initial positions used in Figure 21.

We performed the same robustness tests for the fuzzy and BP-trained truck-and-trailer controllers as in the trailerless truck case. Figure 23 shows performance errors averaged

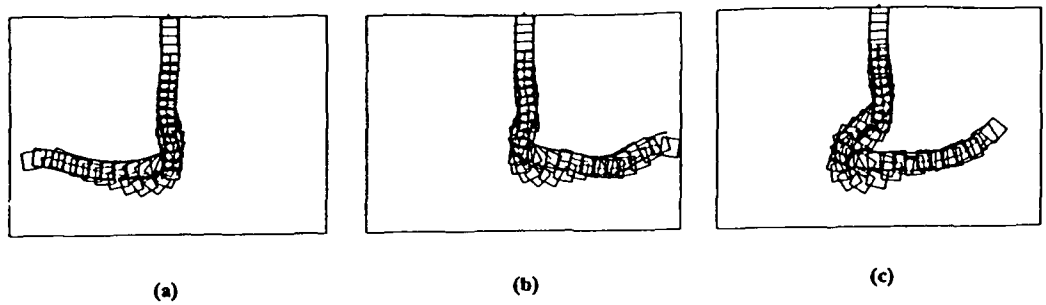


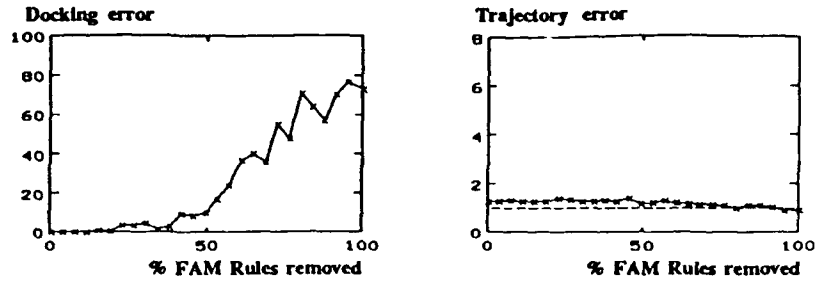
FIGURE 22 Sample truck-and-trailer trajectories of the BP-trained controller for initial positions (x, y, ϕ_c, ϕ_t): (a) (25, 30, −20, 30), (b) (80, 30, 210, −40), and (c) (70, 30, 200, 30).

over ten typical back-ups from ten different initial positions. These performance graphs resemble closely the performance graphs for the trailerless truck systems in Figure 12.

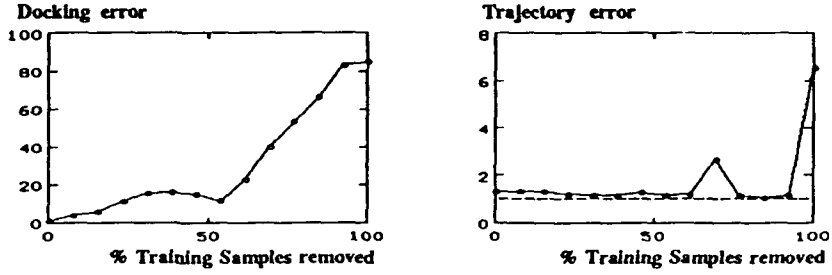
AFAM Truck-and-Trailer Control Systems

We generated 6250 truck-and-trailer data using the original FAM system in Figure 20. We backed up the truck-and-trailer from the same initial positions as in the trailerless truck case. The trailer angle ϕ_t ranged from $-60°$ to $240°$, and the cab angle ϕ_c assumed only the three values $-45°$, $0°$, and $45°$. The training vectors $(x, \phi_t, \phi_c, \beta)$ defined points in the four-dimensional input-output product-space. We nonuniformly partitioned the product space into FAM cells to allow narrower fuzzy-set values near the steady-state FAM cell.

We used DCL to train the AFAM truck-and-trailer controller. The total number of FAM cells equaled 735 ($7 \times 5 \times 7 \times 3$). We used 735 synaptic quantization vectors. The DCL algorithm classified the 6250 data into 105 FAM cells. Figure 24 shows the synaptic-vector histogram corresponding to the 105 FAM rules. Figure 25 shows the estimated FAM bank by the DCL algorithm. Figure 26 shows the original and DCL-estimated control surfaces for the fuzzy truck-and-trailer systems.



(a) Fuzzy truck-and-trailer



(b) BP-Neural truck-and-trailer

FIGURE 23 Comparison of robustness of the two truck-and-trailer controllers: (a) Docking and trajectory error of the fuzzy controller, (b) Docking and trajectory error of the BP controller.

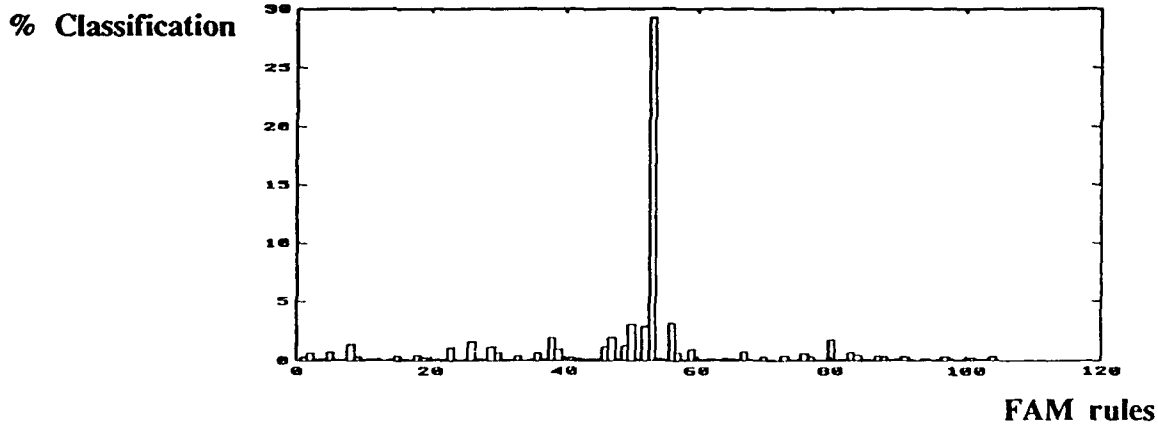


FIGURE 24 Synaptic-vector histogram for the AFAM truck-and-trailer system.

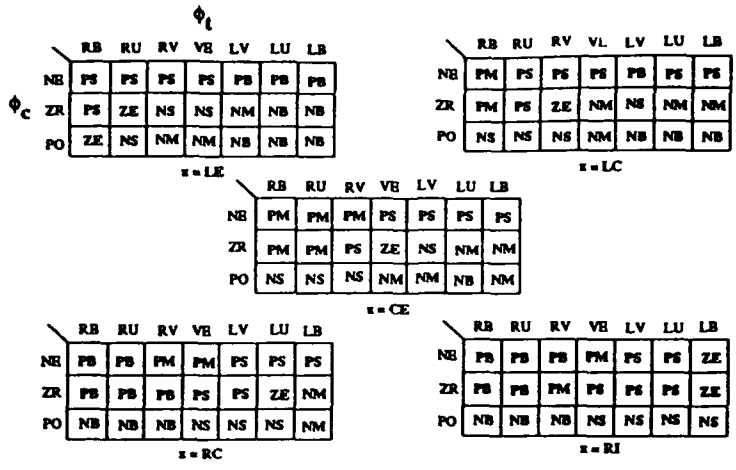
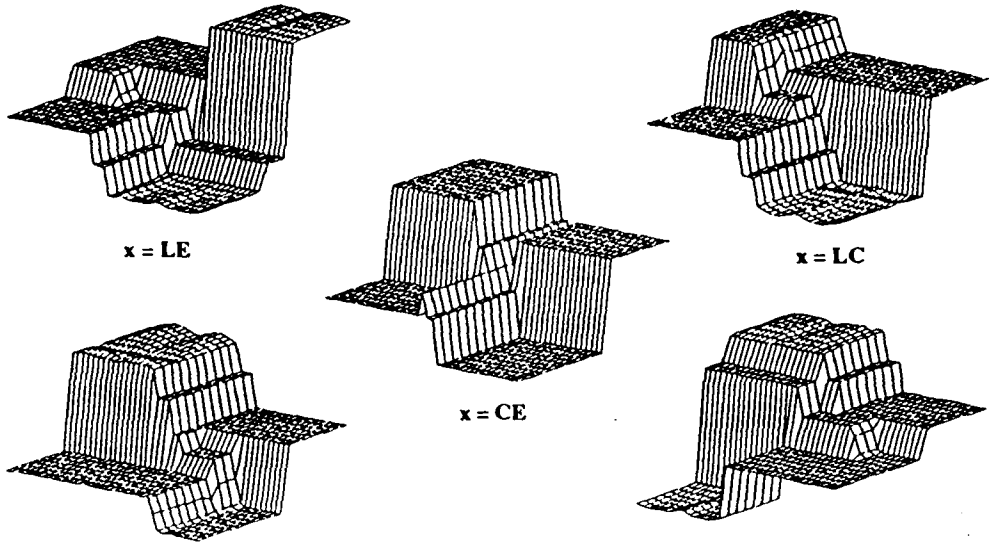
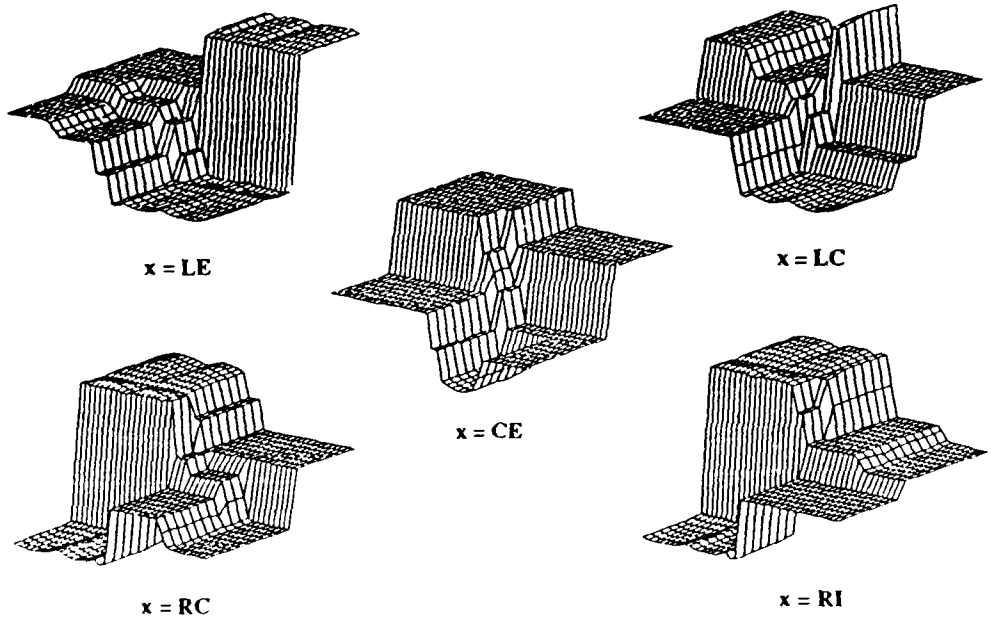


FIGURE 25 DCL-estimated FAM bank for the AFAM truck-and-trailer system.

Figure 27 shows the trajectories of the original FAM and the DCL-estimated AFAM truck-and-trailer controllers. Figure 27a and 27b show the two trajectories from the initial position $(x,\ y,\ \phi_t,\ \phi_c) = (30,30,10,45)$. Figure 27c and 27d show the trajectories from initial position (60,30,210,−60). The original FAM and DCL-estimated AFAM systems exhibited comparable truck-and-trailer control performance except in a few cases, where the DCL-estimated AFAM trajectories were irregular.

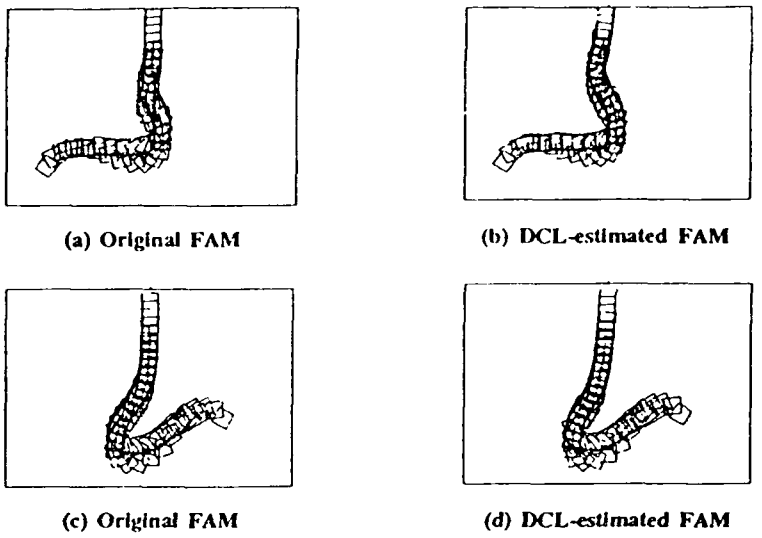


(a) Original control surfaces for the truck-and-trailer system



(b) DCL-estimated control surfaces for the truck-and-trailer system

FIGURE 26 (a) Original control surface (b) DCL-estimated control surface



(a) Original FAM (b) DCL-estimated FAM

(c) Original FAM (d) DCL-estimated FAM

FIGURE 27 Sample truck-and-trailer trajectories from the original and the DCL-estimated FAM systems starting at initial positions $(x, y, \phi_t, \phi_c) =$ (30,30,10,45) and (60,30,210,−60).

Conclusion

We quickly engineered fuzzy systems to successfully back up a truck and truck-and-trailer system in a parking lot. We used only common sense and error-nulling intuitions to generate sufficient banks of FAM rules. These systems performed well until we removed over 50 % of the FAM rules. This extreme robustness suggests that, for many estimation and control problems, different fuzzy engineers can rapidly develop prototype fuzzy systems that perform similarly and well.

The speed with which the DCL clustering technique recovers the underlying FAM bank further suggests that we can likewise construct fuzzy systems for more complex, higher-dimensional problems. For these problems we may have access to only incomplete numerical input-output data. Pure neural-network or statistical-process-control approaches may generate systems with comparable performance. But these systems will involve far greater computational effort, will be more difficult to modify, and will not provide a structured

representation of the system's throughput.

Our neural experiments suggests that whenever we model a system with a neural network, for little extra computational cost we can generate a set of structured FAM rules that approximate the neural system's behavior. We can then tune the fuzzy system by refining the FAM-rule bank with fuzzy-engineering rules of thumb and with further training data.

Acknowledgment

This research was supported by the Air Force Office of Scientific Research (AFOSR-88-0236) and by a grant from the Rockwell Science Center.

References

Huber, P.J., *Robust Statistics*, Wiley, 1981.

Kong, S.G. and Kosko, B., "Differential Competitive Learning for Centroid Estimation and Phoneme Recognition," *IEEE Transactions on Neural Networks*, to appear, January 1991.

Kosko, B., "Fuzzy Entropy and Conditioning," *Information Sciences*, vol.40, 165-174, 1986.

Kosko, B., *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*, Prentice-Hall, 1990.

Kosko, B., "Stochastic Competitive Learning," *Proceedings of the Summer 1990 International Joint Conference on Neural Networks (IJCNN-90)*, vol.II, 215-226, June 1990.

Kosko, B., "Unsupervised Learning in Noise," *IEEE Transactions on Neural Networks*, vol.1, no.1, 44-57, March 1990.

Nguyen, D. and Widrow, B., "The Truck Backer-Upper: An Example of Self-Learning in Neural Networks," *Proceedings of International Joint Conference on Neural Networks (IJCNN-89)*, vol.II, 357-363, June 1989.

APPENDIX: Product-space Clustering with Differential Competitive Learning

Product-space clustering [Kosko, 1990a] is a form of stochastic adaptive vector quantization. Adaptive vector quantization (AVQ) systems adaptively quantize pattern clusters in R^n. Stochastic competitive learning systems are neural AVQ systems. Neurons compete for the activation induced by randomly sampled patterns. The corresponding synaptic fan-in vectors adaptively quantize the pattern space R^n. The p synaptic vectors $\mathbf{m}_j$ define the p columns of the synaptic connection matrix M. M interconnects the n input or linear neurons in the input neuronal field F_X to the p competing nonlinear neurons in the output field F_Y. Figure 28 below illustrates the neural network topology.

Learning algorithms estimate the unknown probability density function $p(\mathbf{x})$, which describes the distribution of patterns in R^n. More synaptic vectors arrive at more probable regions. Where sample vectors $\mathbf{x}$ are dense or sparse, synaptic vectors $\mathbf{m}_j$ should be dense or sparse. The local count of synaptic vectors then gives a nonparametric estimate of the volume probability $P(V)$ for volume $V \subset R^n$:

$$P(V) = \int_V p(\mathbf{x})\, d\mathbf{x} \tag{15}$$

$$\approx \frac{\text{Number of } \mathbf{m}_j \in V}{p} \quad . \tag{16}$$

In the extreme case that $V = R^n$, this approximation gives $P(V) = p/p = 1$. For improbable subsets V, $P(V) = 0/p = 0$.

Stochastic Competitive Learning Algorithms

The metaphor of competing neurons reduces to nearest-neighbor classification. The AVQ system compares the current vector random sample $\mathbf{x}(t)$ in Euclidean distance to the p columns of the synaptic connection matrix M, to the p synaptic vectors $\mathbf{m}_1(t), \ldots, \mathbf{m}_p(t)$. If the jth synaptic vector $\mathbf{m}_j(t)$ is closest to $\mathbf{x}(t)$, then the jth output neuron "wins" the competition for activation at time t. In practice we sometimes define the nearest N synaptic vectors as winners. Some scaled form of $\mathbf{x}(t) - \mathbf{m}_j(t)$ updates the nearest or "winning" synaptic vectors. "Losers" remain unchanged: $\mathbf{m}_i(t+1) = \mathbf{m}_i(t)$. Competitive synaptic vectors converge to pattern-class centroids exponentially fast [Kosko, 1990b].

The following three-step process describes the competitive AVQ algorithm, where the third step depends on which learning algorithm updates the winning synaptic vectors.

Competitive AVQ Algorithm

1. **Initialize synaptic vectors:** $\mathbf{m}_i(0) = \mathbf{x}(i), \quad i = 1, \ldots, p.$
   Sample-dependent initialization avoids many pathologies that can distort nearest-neighbor learning.

2. For random sample $\mathbf{x}(t)$, find the closest or "winning" synaptic vector $\mathbf{m}_j(t)$:

$$\|\mathbf{m}_j(t) - \mathbf{x}(t)\| = \min_i \|\mathbf{m}_i(t) - \mathbf{x}(t)\| \quad , \tag{17}$$

   where $\|\mathbf{x}\|^2 = x_1^2 + \cdots + x_n^2$ defines the squared Euclidean vector norm of $\mathbf{x}$. We can define the N synaptic vectors closest to $\mathbf{x}$ as "winners".

3. Update the winning synaptic vector(s) $\mathbf{m}_j(t)$ with an appropriate learning algorithm.

Differential competitive learning (DCL)

Differential competitive "synapses" learn only if the competing "neuron" changes its competitive status [Kosko, 1990c]:

$$\dot{m}_{ij} = \dot{S}_j(y_j)\,[\,S_i(x_i) - m_{ij}\,] \quad , \tag{18}$$

or in vector notation,

$$\dot{\mathbf{m}}_j = \dot{S}_j(y_j)\,[\,\mathbf{S}(\mathbf{x}) - \mathbf{m}_j\,] \quad , \tag{19}$$

where $\mathbf{S}(\mathbf{x}) = (S_1(x_1), \ldots, S_n(x_n))$ and $\mathbf{m}_j = (m_{1j}, \ldots, m_{nj})$. m_{ij} denotes the synaptic weight between the ith neuron in input field F_X and the jth neuron in competitive field F_Y. Nonnegative signal functions S_i and S_j transduce the real-valued activations x_i and y_j into bounded monotone nondecreasing signals $S_i(x_i)$ and $S_j(y_j)$. $\dot{m}_{ij}$ and $\dot{S}_j(y_j)$ denote the time derivatives of m_{ij} and $S_j(y_j)$, synaptic and signal velocities. $S_j(y_j)$ measures the competitive status of the jth competing neuron in F_Y. Usually S_j approximates a binary threshold function. For example, S_j may equal a steep binary logistic sigmoid,

$$S_j(y_j) = \frac{1}{1 + e^{-cy_j}} \quad , \tag{20}$$

for some constant $c > 0$. The jth neuron wins the laterally inhibitive competition if $S_j = 1$, loses if $S_j = 0$.

For discrete implementation, we use the DCL algorithm as a stochastic difference equation [Kong, 1991]:

$$\mathbf{m}_j(t+1) = \mathbf{m}_j(t) + c_t\,\Delta S_j(y_j(t))\,[\,S(\mathbf{x}(t)) - \mathbf{m}_j(t)\,] \text{ if the } j\text{th neuron wins,} \tag{21}$$

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) \qquad \text{if the } i\text{th neuron loses.} \tag{22}$$

$\Delta S_j(y_j(t))$ denotes the time change of the jth neuron's competition signal $S_j(y_j)$ in the competitive field F_Y:

$$\Delta S_j(y_j(t)) = \mathrm{sgn}[\,S_j(y_j(t+1)) - S_j(y_j(t))\,] \quad . \tag{23}$$

We define the signum operator $\mathrm{sgn}(x)$ as

$$\mathrm{sgn}(x) = \begin{cases} 1 & \text{if} \quad x > 0 \\ 0 & \text{if} \quad x = 0 \\ -1 & \text{if} \quad x < 0 \end{cases} \quad . \tag{24}$$

$\{c_t\}$ denotes a slowly decreasing sequence of learning coefficients, such as $c_t = .1\,(1 - t/2000)$ for 2000 training samples. Stochastic approximation [Huber, 1981] requires a decreasing gain sequence $\{c_t\}$ to suppress random disturbances and to guarantee convergence to local minima of mean-squared performance measures. The learning coefficients should decrease slowly,

$$\sum_{t=1}^{\infty} c_t = \infty \quad , \tag{25}$$

but not too slowly,

$$\sum_{t=1}^{\infty} c_t^2 < \infty \quad . \tag{26}$$

Harmonic-series coefficients, $c_t = 1/t$, satisfy these constraints.

We approximate the competitive signal difference ΔS_j as the activation difference Δy_j:

$$\Delta S_j(y_j(t)) = \mathrm{sgn}[\, y_j(t+1) - y_j(t) \,] \tag{27}$$

$$= \Delta y_j(t) \quad . \tag{28}$$

Input neurons in feedforward networks usually behave linearly: $S_i(x_i) = x_i$, or $S(\mathbf{x}(t)) = \mathbf{x}(t)$. Then we update the winning synaptic vector $\mathbf{m}_j(t)$ with

$$\mathbf{m}_j(t+1) = \mathbf{m}_j(t) + c_t\, \Delta y_j(t)\, [\, \mathbf{x}(t) - \mathbf{m}_j(t) \,] \quad . \tag{29}$$

We update the F_Y neuronal activations y_j with the additive model

$$y_j(t+1) = y_j(t) + \sum_{i}^{n} S_i(x_i(t))\, m_{ij}(t) + \sum_{k}^{p} S_k(y_k(t))\, w_{kj} \quad . \tag{30}$$

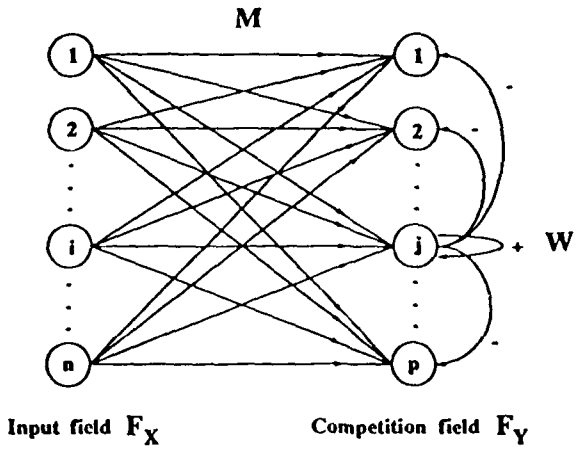


FIGURE 28 Topology of the laterally inhibitive DCL network.

For linear signal functions S_i, the first sum in (30) reduces to an inner product of sample and synaptic vectors:

$$\sum_{i}^{n} x_i(t)\, m_{ij}(t) \;=\; \mathbf{x}^T(t)\, \mathbf{m}_j(t) \quad . \tag{31}$$

Then positive learning tends to occur—$\Delta m_{ij} > 0$—when $\mathbf{x}$ is close to the jth synaptic vector $\mathbf{m}_j$.

Since a binary threshold function approximates the output signal function $S_k(y_k)$, the second sum in (30) sums over just the winning neurons: $\sum_k w_{kj}$ for all winning neurons y_k .

The $p \times p$ matrix W contains the F_Y within-field synaptic connection strengths. Diagonal elements w_{ii} are positive, off-diagonal elements negative. Winning neurons excite themselves and inhibit all other neurons. Figure 28 shows the connection topology of the laterally inhibitive DCL network.

Product-space clustering

We divided the space $0 \leq x \leq 100$ into five nonuniform intervals $[0, 32.5]$, $[32.5, 47.5]$, $[47.5, 52.5]$, $[52.5, 67.5]$, and $[67.5, 100]$. Each interval represented the five fuzzy-set values LE, LC, CE, RC, and RI. This choice corresponded to the nonoverlapping intervals of the fuzzy membership function graphs $m(x)$ in Figure 3. Similarly, we divided the space $-90 \leq \phi \leq 270$ into seven nonuniform intervals $[-90, 0]$, $[0, 66.5]$, $[66.5, 86]$, $[86, 94]$, $[94, 113.5]$, $[113.5, 182.5]$, and $[182.5, 270]$, which corresponded respectively to RB, RU, RV, VE, LV, LU, and LB. We divided the space $-30 \leq \theta \leq 30$ into seven nonuniform intervals $[-30, -20]$, $[-20, -7.5]$, $[-7.5, -2.5]$, $[-2.5, 2.5]$, $[2.5, 7.5]$, $[7.5, 20]$, and $[20, 30]$, which corresponded to NB, NM, NS, ZE, PS, PM, and PB.

DCL classified each input-output data vector into one of the FAM cells. We added a FAM rule to the FAM bank if the DCL-trained synaptic vector fell in the FAM cell. In case of ties we chose the FAM cell with the most densely clustered data.

For the BP-AFAM generated from the neural control surface in Figure 15, we divided the rectangle $[0, 100] \times [-90, 270]$ into 35 nonuniform squares with the same divisions defined above. Then we added and averaged the control surface values in the square. We added a FAM rule to the FAM bank if the averaged value corresponded to one of the seven FAM cells.

For the truck-and-trailer case, we divided the space $-90 \leq \phi_c \leq 90$ into three intervals $[-90, -12.5]$, $[-12.5, 12.5]$, and $[12.5, 90]$, which corresponded to NE, ZR, and PO. There were 735 FAM cells, and 735 possible FAM rules, of the form $(x, \phi_t, \phi_c; \beta)$.

Differential Competitive Learning for Centroid Estimation and Phoneme Recognition

SEONG-GON KONG AND BART KOSKO

NN/December 1990 IEEE

Abstract—We compared a differential competitive learning (DCL) system with two supervised competitive learning (SCL) systems for centroid estimation and phoneme recognition. DCL provides a new form of unsupervised adaptive vector quantization. Standard stochastic competitive learning systems learn only if neurons win a competition for activation induced by randomly sampled patterns. DCL systems learn only if the competing neurons *change* their competitive signal. Signal-velocity information provides unsupervised *local reinforcement* during learning. The sign of the neuronal signal derivative rewards winners and punishes losers. Standard competitive learning ignores instantaneous *win-rate* information. Synaptic fan-in vectors adaptively quantize the randomly sampled pattern space into nearest-neighbor decision classes. More generally, the synaptic vector distribution estimates the unknown sampled probability density function $p(x)$. Simulations showed that unsupervised DCL-trained synaptic vectors converged to class centroids at least as fast as, and wandered less about these centroids than, SCL-trained synaptic vectors. Simulations on a small set of English phonemes favored DCL over SCL for classification accuracy.

I. Adaptive Vector Quantization for Phoneme Recognition

PHONEME recognition is a simple form of speech recognition. We can recognize a speech sample phoneme by phoneme. The phoneme recognition system learns only a comparatively small set of minimal syllables or phonemes. More advanced systems learn and recognize words, phrases, or sentences. There are orders of magnitude more such speech units than phonemes. Words and phrases can also undergo more complex forms of distortion and time warping.

In principle, we can recognize phonemes and speech with *vector quantization* methods. These methods search for a small but representative set of prototypes, which we can then use to match sample patterns with nearest-neighbor techniques.

In neural network phoneme recognition, a sequence of discrete phonemes from a continuous speech sample produces a series of neuronal responses. Kohonen's [4] supervised neural phoneme recognition system successfully classifies 21 Finnish phonemes. This stochastic competitive learning system behaves as an adaptive vector quantization system.

Manuscript received February 22, 1990; revised July 24, 1990. This paper was supported by the Air Force Office of Scientific Research under Grant AFOSR-88 0236 and by a grant from Nippon Telephone and Telegraph.

The authors are with the Department of Electrical Engineering, Signal and Image Processing Institute, University of Southern California, Los Angeles, CA 90089-0272.

IEEE Log Number 9038994

Traditional vector quantization systems may attempt to minimize a mean-squared-error or entropic performance measure. Formal minimization assumes knowledge of the sampled probability density function $p(x)$ and perhaps additional knowledge of how some parameters functionally depend on other parameters. $p(x)$ describes the continuous distribution of patterns in R^n. In general, we do not know this probabilistic information. Instead, we use learning algorithms to adaptively estimate $p(x)$ from sample realizations. This procedure often reduces to *stochastic approximation* [11], [12].

Adaptive vector quantization (AVQ) systems adaptively quantize pattern clusters in R^n. Stochastic competitive learning systems are neural AVQ systems. Neurons compete for the activation induced by randomly sampled patterns. The corresponding synaptic fan-in vectors adaptively quantize the pattern space R^n. The p synaptic vectors m_j define the p columns of the synaptic connection matrix M. M interconnects the n input or linear neurons in the input neuronal field F_X to the p competing nonlinear neurons in the output field F_Y.

In the simplest case, the p synaptic vectors estimate centroids or modes of the sampled probability density function $p(x)$. The estimates are nonparametric. The user need not know or assume which probability density function $p(x)$ generates the training samples, the observed realizations of the underlying stochastic pattern process.

Pattern learning is *supervised* if the system uses pattern-class information. Suppose the k decision classes $\{D_j\}$ partition the pattern space R^n:

$$R^n = \bigcup_{j=1}^{k} D_j \quad \text{and} \quad D_i \cap D_j = \emptyset \quad \text{if } i \neq j. \tag{1}$$

The system knows and uses the class membership of each pattern x. The system knows that $x \in D_i$ and that $x \notin D_j$ for all $j \neq i$. Pattern learning is *unsupervised* if the system does not know or use class membership information. Unsupervised learning algorithms use *unlabeled* pattern samples.

Formally supervised learning depends on class *indicator functions* $\{I_{D_j}\}$:

$$I_{D_j}(x) = \begin{cases} 1 & \text{if } x \in D_j \\ 0 & \text{if } x \notin D_j. \end{cases} \tag{2}$$

I_{D_j} indicates whether pattern x belongs to decision class D_j. Unsupervised learning algorithms blindly cluster samples. They do not depend on class indicator functions. The random indicator functions define the *class probabilities* $P(D_1), \cdots, P(D_k)$, since

$$P(D_j) = \int_{D_j} p(x)\,dx \tag{3}$$

$$= \int_{R^n} I_{D_j}(x)\,p(x)\,dx \tag{4}$$

$$= E[I_{D_j}]. \tag{5}$$

$E[x]$ denotes the mathematical expectation of scalar random variable x. The partition property and $P(R^n) = 1$ imply $P(D_1) + \cdots + P(D_k) = 1$.

Learning algorithms estimate the unknown probability density function $p(x)$. We need not learn if we know $p(x)$. Instead, we could compute the desired quantities with optimization, numerical-analytical, or calculus-of-variation techniques. For instance, we could directly compute the centroids $\bar{x}_j$ of the pattern classes D_j. The centroids minimize the total mean-squared-error of vector quantization $\mathcal{E}$,

$$\mathcal{E} = \tfrac{1}{2} \sum_{j}^{k} \int_{D_j} \sum_{i}^{n} (x_i - m_{ij})^2\,p(x)\,dx. \tag{6}$$

For if we set the gradient vector $\nabla_{m_j}\mathcal{E}$ equal to the null vector and solve for the optimal $\hat{m}_j$, we get

$$\mathbf{0} = \nabla_{m_j}\mathcal{E} \tag{7}$$

$$= \int_{D_j} (x - \hat{m}_j)\,p(x)\,dx \tag{8}$$

$$= \int_{D_j} xp(x)\,dx - \hat{m}_j \int_{D_j} p(x)\,dx. \tag{9}$$

Then

$$\hat{m}_j = \frac{\int_{D_j} xp(x)\,dx}{\int_{D_j} p(x)\,dx} \tag{10}$$

$$= \bar{x}_j\,, \tag{11}$$

as claimed (when positive-definite Hessian conditions hold).

Mean-squared-error optimal learning drives synaptic vectors to the unknown centroids $\bar{x}_j$ of the locally sampled pattern classes. More generally [8], $E[m_j] = \bar{x}_j$ holds asymptotically as the random synaptic vector m_j wanders in a Brownian motion about the centroid $\bar{x}_j$. We observed this Brownian wandering in the simulations discussed below (Fig. 6).

If there are exactly p distinct pattern classes or clusters, the p synaptic row vectors $m_1(t), \cdots, m_p(t)$ should asymptotically approach the centroid of a distinct pattern class. In general, we do not know the number k of pattern classes. If there are fewer synaptic vectors than the number k of pattern classes, if $p < k$, the synaptic vectors should approach the centroids of the p most massive, most probable pattern clusters.

If $p > k$, the synaptic vectors should approximate the entire density function $p(x)$. More synaptic vectors should arrive at more probable regions. Where patterns x are dense or sparse, synaptic vectors m_j should be dense or sparse. The local count of synaptic vectors then gives an accurate nonparametric estimate of the volume probability $P(V)$ for volume $V \subset R^n$:

$$P(V) = \int_V p(x)\,dx \tag{12}$$

$$\approx \frac{\text{number of } m_j \in V}{p}\,. \tag{13}$$

In the extreme case that $V = R^n$, this approximation gives $P(V) = p/p = 1$. For small or improbable subsets V, $P(V) = 0/p = 0$.

Differential competitive learning (DCL) provides a new [7] unsupervised form of AVQ. DCL modifies stochastic synaptic vectors with a competing neuron's *change* in output signal. The neuronal signal velocity locally reinforces the synaptic vector. The time derivative's sign changes resemble the supervised sign changes in supervised competitive learning (SCL) algorithms. SCL systems use more information than DCL systems, since signal velocities do not depend on class-membership information. In particular, the DCL algorithm in (40) below does not use the class membership of the training sample x.

Both DCL-trained and SCL-trained synaptic vectors tend to rapidly converge to pattern-class centroids [8]. Our simulated DCL synaptic vectors converged faster to bipolar centroids—points in $\{-1, 1\}^n$—than did SCL synaptic vectors when, as in biological neural networks, a sigmoidal signal function nonlinearly transduced neuronal activations to bounded signals. DCL systems exploit a *win-rate* dependent sequence of learning coefficients. The faster the neuron wins or loses, the more the synaptic vector resembles or disresembles the sampled pattern. SCL systems ignore this instantaneous rate information.

In practice, input neurons have linear signal functions: $S_i(x_i) = x_i$. The user presents the random sample x to the system as the *output* of the F_X neurons. In this case, our simulated DCL and SCL synaptic vectors converged equally quickly. But the DCL-trained synaptic vectors wandered less about class centroids than did SCL-trained synaptic vectors.

II. Stochastic Competitive Learning Algorithms

Autoassociative AVQ neural networks are two-layer feedforward networks trained with competitive learning. The input neuronal field F_X receives the sample data and passes it forward through synaptic connection matrix M to the p competing neurons in field F_Y. (Heteroassociative AVQ networks correspond to three-layer feedforward networks.) Synchronous feedforward flow obviates the neural interpretation. AVQ neural systems are simply signal processing algorithms.

The metaphor of competing neurons reduces to a nearest-neighbor classification. The system compares the current vector random sample $x(t)$ in Euclidean distance to the p columns of the synaptic connection matrix M, to the p synaptic vectors $m_1(t), \cdots, m_p(t)$. If the jth synaptic vector $m_j(t)$ is closest to $x(t)$, then the jth neuron "wins" the competition for activation at time t.

Many within-field feedback dynamical systems approximate this nearest-neighbor, winner-take-all behavior. Mathematically, the jth competing neuron should behave as a class indicator function: $S_j = I_{D_j}$. More generally, the jth F_Y neuron only estimates I_{D_j}. Then misclassification can still occur: $S_j(xm_j^T + f_j) = 1$ but $I_{D_j}(x) = 0$ for row vectors x and m_j, where f_j denotes the inhibitive within-field feedback the jth neuron receives.

We modify the nearest or "winning" synaptic vector m_j with a simple difference learning law. We add some scaled form of $x(t) - m_j(t)$ to $m_j(t)$ to form $m_j(t + 1)$. We can also update near-neighbors of the winning neuron. In practice, and in the simulations below, we modify only one synaptic vector at a time. We do not modify "losers": $m_i(t + 1) = m_i(t)$.

The stochastic *unsupervised competitive learning* (UCL) algorithm represents the simplest competitive learning algorithm. Pattern recognition theorists first studied the UCL algorithm but called it *adaptive K-means clustering* [10]. Kohonen extended the UCL algorithm to two supervised versions, SCL1 [3], [4] and SCL2 [5]. **The supervisor must know the class membership of each sample pattern x. The SCL1 and SCL2 algorithms linearly "reward" correct classifications as in the UCL algorithm. They "punish" incorrect classifications with a sign change. We obtain all three algorithms from the following three-step algorithm if we replace the third step with the appropriate stochastic difference equation.**

A. Competitive AVQ Algorithms

1) Initialize synaptic vectors: $m_i(0) = x(i)$, $i = 1, \cdots, p$. Sample-dependent initialization avoids many pathologies that can distort nearest-neighbor learning.

2) For random sample $x(t)$, find the closest or "winning" synaptic vector $m_j(t)$:

$$\|m_j(t) - x(t)\| = \min_i \|m_i(t) - x(t)\|, \quad (14)$$

where $\|x\|^2 = x_1^2 + \cdots + x_n^2$ defines the squared Euclidean vector norm of x.

3) Update the winning synaptic vector $m_j(t)$ with the UCL, SCL1, or SCL2 learning algorithm.

B. Unsupervised Competitive Learning (UCL)

$$m_j(t + 1) = m_j(t) + c_t[x(t) - m_j(t)], \quad (15)$$

$$m_i(t + 1) = m_i(t) \quad \text{if } i \neq j, \quad (16)$$

where $\{c_t\}$ denotes a slowly decreasing sequence of learning coefficients. In our simulations, $c_t = 0.1(1 - (t/2000))$ for 2000 training samples. The UCL algorithm (15) restates the classical adaptive K-means clustering algorithm.

Stochastic approximation [11] requires a decreasing gain sequence $\{c_t\}$ to suppress random disturbances and to guarantee convergence to a local minima of mean-squared performance measures. The learning coefficients should decrease slowly,

$$\sum_{t=1}^{\infty} c_t = \infty, \quad (17)$$

but not too slowly,

$$\sum_{t=1}^{\infty} c_t^2 < \infty. \quad (18)$$

Harmonic-series coefficients, $c_t = 1/t$, satisfy these constraints. For fast robust [2] stochastic approximation, only the harmonic-series coefficients satisfy these constraints.

C. Supervised Competitive Learning 1 (SCL1)

$$m_j(t + 1) = \begin{cases} m_j(t) + c_t[x(t) - m_j(t)] \\ \quad \text{if } x(t) \in D_j \\ m_j(t) - c_t[x(t) - m_j(t)] \\ \quad \text{if } x(t) \notin D_j \end{cases}, \quad (19)$$

$$m_i(t + 1) = m_i(t) \quad \text{if } i \neq j. \quad (20)$$

SCL1 supervises or reinforces synaptic modification. m_j learns positively if the system correctly classifies the random sample x. m_j learns negatively, or forgets selectively, if the system misclassifies the random sample. Then m_j tends to move out of regions of misclassification in R^n. Tsypkin [12] first derived the SCL1 algorithm as a special case of his adaptive Bayes classifier.

We can rewrite the SCL1 update equation (19) as

$$m_j(t + 1) = m_j(t) + c_t r_j(x(t))[x(t) - m_j(t)] \quad (21)$$

if we define the supervised *reinforcement function* r_j as

$$r_j = I_{D_j} - \sum_{i \neq j} I_{D_i}. \quad (22)$$

r_j depends explicitly on class indicator functions. r_j rewards correct pattern classifications with $+1$ and punishes misclassifications with -1. We implicitly assume the jth neuron accurately estimates the jth indicator function: $S_j(xm_j^T + f_j) \approx I_{D_j}(x)$.

The SCL2 algorithm slightly modifies the SCL1 algorithm. The SCL2 algorithm better estimates the optimal Bayes decision-theoretic boundary in some cases. The Bayes decision boundary minimizes the misclassification error. It represents the crossing point of the unknown conditional densities $p(x \mid D_i)$ and $p(x \mid D_j)$.

The nearest-neighbor decision boundary corresponds to the hyperplane that bisects the line that connects the two class centroids. If the pattern distribution is asymmetric—if, for instance, local density functions with different variances generate different decision classes—then the SCL1 decision boundary may not resemble the Bayes decision boundary. Nearest-neighbor classification tends to perform better in the equal variance case than in the unequal variance case.

D. Supervised Competitive Learning 2 (SCL2)

$$m_j(t+1) = m_j(t) - c_t[x(t) - m_j(t)] \quad , \quad (23)$$

$$m_I(t+1) = m_I(t) + c_t[x(t) - m_I(t)] \quad , \quad (24)$$

if $x \in D_I$ instead of $x \in D_j$, and if $m_j(t)$ is the nearest synaptic vector and $m_I(t)$ is the next-to-nearest synaptic vector :

$$\|m_j(t) - x(t)\| < \|m_I(t) - x(t)\| = \min_{i \neq j} \|m_i(t) - x(t)\| \quad , \quad (25)$$

and if $x(t)$ falls in a class-dependent "window." In all other cases,

$$m_i(t+1) = m_i(t). \quad (26)$$

The window defines a hyperrectangle in R^n centered at the midpoint of the hyperline that connects the centroids of D_j and D_I. If $x(t)$ does not fall in the hyperwindow, we modify no synaptic vector. We defined the R^n window between D_j and D_I as the n-dimensional hyperrectangle $[\overline{m}_1 - d, \overline{m}_1 + d] \times \cdots \times [\overline{m}_n - d, \overline{m}_n + d]$, where $\overline{m}_{jI}$ denotes the midpoint $\overline{m}_{jI} = (\overline{m}_1, \cdots, \overline{m}_n) = (m_j + m_I)/2$, and d denotes the window half-width. We put $d = 2.5$.

III. Differential Competitive Learning

The *differential competitive learning* (DCL) law [7] combines competitive and differential Hebbian learning:

$$\dot{m}_{ij} = \dot{S}_j(y_j)[S_i(x_i) - m_{ij}] \quad , \quad (27)$$

or in vector notation,

$$\dot{m}_j = \dot{S}_j(y_j)[S(x) - m_j] \quad , \quad (28)$$

where $S(x) = (S_1(x_1), \cdots, S_n(x_n))$ and $m_j = (m_{1j}, \cdots, m_{nj})$. m_{ij} denotes the synaptic weight between the ith neuron in input neuronal field F_X and the jth neuron in competitive field F_Y. Nonnegative *signal functions* S_i and S_j transduce the real-valued *activations* x_i and y_j into the bounded monotone nondecreasing *signals* $S_i(x_i)$ and $S_j(y_j)$. $\dot{m}_{ij}$ and $\dot{S}_j(y_j)$ denote the time derivatives of m_{ij} and $S_j(y_j)$, synaptic and signal velocities.

The stochastic calculus version of the DCL law relates random processes :

$$dm_{ij} = dS_j[S_i - m_{ij}] + dB_{ij}. \quad (29)$$

B_{ij} denotes a Brownian-motion diffusion process centered at the origin. We can rewrite (29) in "noise" notation as

$$\dot{m}_{ij} = \dot{S}_j[S_i - m_{ij}] + n_{ij}. \quad (30)$$

The "noise" process n_{ij} has zero mean, $E[n_{ij}] = 0$, and has finite variance, $V[n_{ij}] = \sigma_{ij}^2 < \infty$. The random-sampling AVQ framework implicitly assumes that all competitive learning laws are stochastic differential or difference equations. Such stochastic synaptic vectors m_j tend to converge to pattern-class centroids, and converge exponentially quickly [8].

$S_j(y_j)$ measures the competitive status of the jth competing neuron in F_Y. Usually, S_j approximates a binary threshold function. S_j may equal a steep binary logistic sigmoid,

$$S_j(y_j) = \frac{1}{1 + e^{-cy_j}} \quad , \quad (31)$$

for some constant $c > 0$. The jth neuron wins the laterally inhibitive competition if $S_j = 1$, loses if $S_j = 0$.

In (27), m_j learns only if $S_j(y_j)$ changes. This contrasts with the classical competitive learning law

$$\dot{m}_{ij} = S_j(y_j)[S_i(x_i) - m_{ij}] \quad , \quad (32)$$

which modulates the difference $S(x) - m_j$ with the win-loss signal S_j, not its velocity $\dot{S}_j$. In (32), m_j learns only if the competitive signal S_j exceeds zero—only if the jth neuron "wins" the activation competition.

Real neurons transmit and receive pulse trains. Pulse-coded signal functions S_j reveal the connection between competitive and differential competitive learning. A pulse-coded signal function uses an exponentially fading window [1] of sampled binary pulses :

$$S_j(t) = \int_{-\infty}^{t} y_j(s)\, e^{s-t}\, ds \quad , \quad (33)$$

where $y_j(t) = 1$ if a pulse occurs at t, and $y_j(t) = 0$ if no pulse occurs at t. Then [9]

$$\dot{S}_j(t) = y_j(t) - S_j(t). \quad (34)$$

So the DCL law (27) reduces to

$$\dot{m}_{ij} = y_j[S_i - m_{ij}] - S_j[S_i - m_{ij}]. \quad (35)$$

When the second term in (35) is sufficiently small, DCL reduces to competitive learning. This occurs when a losing neuron suddenly wins, for then $y_j = 1$ and $S_j \approx 0$. In the stochastic case, the *random* pulse function y_j represents an arbitrary random point process, and converts (35) to a doubly stochastic model.

Similarly, the classical differential Hebbian law [6]

$$\dot{m}_{ij} = -m_{ij} + \dot{S}_i \cdot \dot{S}_j \quad (36)$$

reduces to signal Hebbian learning on average (in the absence of pulses) :

$$\dot{m}_{ij} = -m_{ij} + \dot{S}_i \dot{S}_j \quad (37)$$

$$= -m_{ij} + S_i S_j + [x_i y_j - x_i S_j - y_j S_i] \quad (38)$$

$$\approx -m_{ij} + S_i S_j \quad , \quad (39)$$

on average. The approximation holds exactly if and only if no x_i or y_j pulses are present, a frequent event. Differential Hebbian learning synapses "fill in" with Hebbian learning when pulses are absent.

For discrete implementation, we use the DCL algorithm as a stochastic difference equation.

A. Differential Competitive Learning (DCL)

1) Initialize: $m_i(0) = x(i)$.

2) Find winning $m_j(t)$: $\|m_j(t) - x(t)\| = \min_i \|m_i(t) - x(t)\|$.

3) Update winning $m_j(t)$:

$$m_j(t+1) = m_j(t) + c_t \Delta S_j(y_j(t)) \left[S(x(t)) - m_j(t)\right] \quad \text{if the } j\text{th neuron wins,} \tag{40}$$

$$m_i(t+1) = m_i(t) \quad \text{if the } i\text{th neuron loses.} \tag{41}$$

$\Delta S_j(y_j(t))$ denotes the time change of the jth neuron's competition signal $S_j(y_j)$ in the competition layer F_Y:

$$\Delta S_j(y_j(t)) = \operatorname{sgn}\left[S_j(y_j(t+1)) - S_j(y_j(t))\right]. \tag{42}$$

We define the *signum operator* sgn (x) as

$$\operatorname{sgn}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0. \end{cases} \tag{43}$$

We update the F_Y neuronal activations y_j with the additive model

$$y_j(t+1) = y_j(t) + \sum_i^n S_i(x_i(t))\, m_{ij}(t) + \sum_k^p S_k(y_k(t))\, w_{kj}. \tag{44}$$

In our simulations, the first sum in (44) reduced to

$$\sum_i^n x_i(t)\, m_{ij}(t) \tag{45}$$

when we did not transform the input patterns x with a nonlinear signal function S_i. Input or F_X neurons in feedforward networks usually behave linearly: $S_i(x_i) = x_i$.

For linear inputs, we computed the second sum in (44) for linear-signal functions S_k. Since we allowed only one winner per iteration, this sum reduced to a single term $y_k w_{kj}$, where k denotes the winning neuron.

The $p \times p$ matrix W defined the F_Y within-field synaptic connection strengths:

$$W = \begin{bmatrix} +2 & -1 & -1 & \cdots & -1 \\ -1 & +2 & -1 & \cdots & -1 \\ & & \vdots & & \\ -1 & -1 & -1 & \cdots & +2 \end{bmatrix}. \tag{46}$$

Diagonal elements w_{ii} equaled 2, off-diagonal elements equaled -1. Fig. 1 shows the connection topology of the laterally inhibitive DCL network.

Each neuron in F_Y codes for a specific pattern class. By (44) and the "cosine law",

$$S(x) \cdot m_j = \|S(x)\|\, \|m_j\| \cos\left(S(x), m_j\right) \tag{47}$$

positive learning ($\dot{m}_{ij} > 0$) tends to occur when the system classifies x to the nearest pattern class D_j.

If we represent the F_X signal function S_i with the bipolar logistic function,

$$S_i(x_i) = \frac{2}{1 + e^{-cx_i}} - 1, \tag{48}$$

$c > 0$, then the DCL algorithm (40) abstracts the corresponding bipolar pattern from the real-valued input. The unsupervised sign change ΔS_j in the DCL law (40) resembles the reward–punish sign change in the SCL1 and SCL2 algorithms. This suggests that we can meaningfully compare the algorithms' performance on the same training and test data.

If we choose $S_i(x_i)$ as a linear function of the input, if $S_i(x_i) = x_i$, then the discrete version of DCL resembles the UCL, SCL1, and SCL2 algorithms. We used both linear and nonlinear formulations to compare DCL to SCL1 and SCL2. The supervised SCL1 and SCL2 algorithms always outperformed the UCL algorithm. So we limited our DCL comparisons to SCL1 and SCL2 systems.

For most simulations, we used linearly transformed data, $S_i(x_i) = x_i$. In these cases, we approximated the signal difference ΔS_j as the activation difference Δy_j:

$$\Delta S_j(y_j(t)) \approx \Delta y_j(t) \tag{49}$$

$$= \operatorname{sgn}\left[y_j(t+1) - y_j(t)\right]. \tag{50}$$

This approximation holds exactly over the linear part of a signal function's range. For then $S_j' = dS_j/dy_j = c$ for some constant $c > 0$. Then

$$\dot{S}_j = S_j' \dot{y}_j \tag{51}$$

$$= c\dot{y}_j. \tag{52}$$

The constant c does not affect the signum operator used in Δy_j.

Linear data often produce large activation sums $\Sigma_i^n x_i m_{ij}$ that saturate nonlinear signals S_j to extreme values. Then the signal difference ΔS_j equals zero and may not discriminate changes in the competitive status. The activation difference Δy_j remains sensitive to these changes.

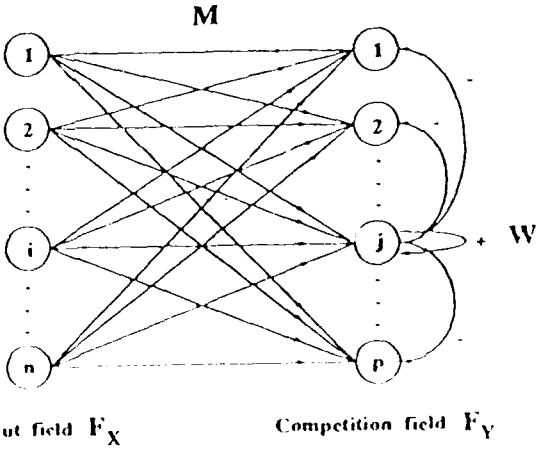


Fig. 1. Topology of the laterally inhibitive DCL network.

IV. Comparison of Competitive and Differential Competitive Learning for Centroid Estimation

We compared the DCL algorithm to the SCL1 and SCL2 algorithms for estimating centroids. All algorithms adaptively moved the synaptic vectors m_j to pattern-class centroids. They differed in how quickly the trained synaptic vectors reached the centroids and how much the synaptic vectors wandered about the centroids. The DCL algorithm moved the synaptic vectors to centroids at least as fast as did the SCL1 and SCL2 algorithms. Once the synaptic vectors reached the pattern-class centroids, the DCL-trained synaptic vectors wandered less about the centroids than the SCL-trained synaptic vectors.

The DCL algorithm converged to centroids faster than the SCL1 and SCL2 algorithms. Convergence rates were the same for linear signal functions, $S_i(x_i) = x_i$. The pattern space consisted of 2000 two-dimensional Gaussian-distributed pattern vectors with variance 121 and with centroids or modes at (20, 20), (20, −20), (−20, 20), and (−20, −20). Fig. 2 shows centroid convergence of DCL synaptic vectors with inputs transformed with bipolar signal functions. Fig. 3 shows the slower convergence of the SCL1 algorithm with the same transformed Gaussian Data. * denotes DCL synaptic vectors. + denotes SCL1 synaptic vectors. Figs. 4 and 5 show centroid convergence for the same Gaussian data when the systems used linear signal functions.

DCL-trained synaptic vectors wandered with less mean-squared error about centroids than did SCL-trained synaptic vectors. Fig. 6 shows mean-squared wandering about the Gaussian pattern-class centroid (−20, 20). Fig. 6 represents several such experiments with different Gaussian and non-Gaussian pattern distributions. Solid lines denote the convergence of the DCL synaptic vector. Dashed lines denote convergence of the SCL1 synaptic vector.

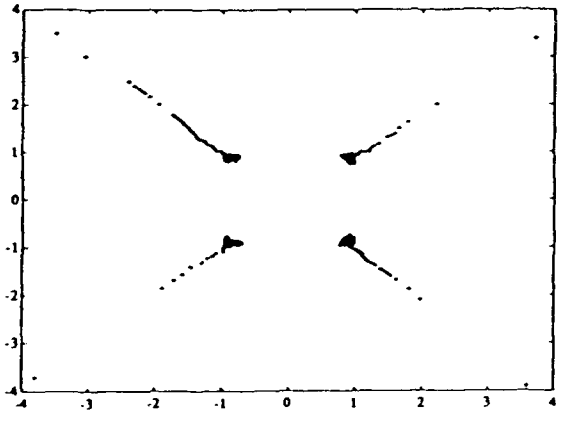


Fig. 2. Convergence of DCL-trained synaptic vectors to bipolar centroids of four Gaussian clusters. Bipolar logistic signal functions $S_i(x_i)$ nonlinearly transduce the real-valued input vector x into a bipolar vector in $\{-1, 1\}^n$.

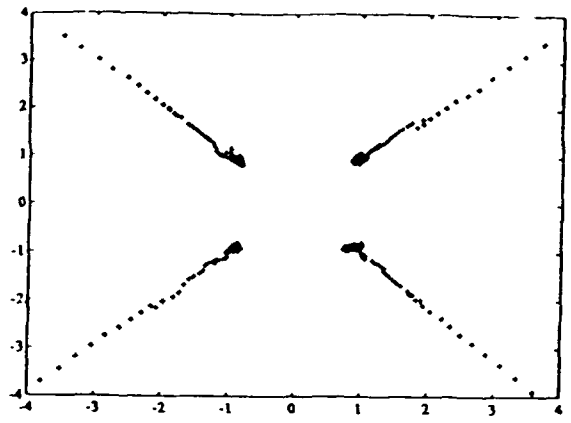


Fig. 3. Centroid convergence of SCL1 synaptic vectors trained with the same patterns as in Fig. 2. Bipolar logistic signal functions nonlinearly transduce real input patterns to bipolar patterns.

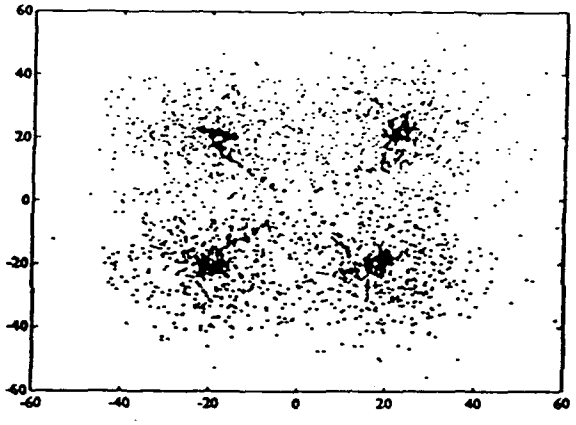


Fig. 4. Convergence of DCL-trained synaptic vectors to Gaussian pattern-class centroids. Same pattern distribution as in Figs. 2 and 3. Input data not transformed: $S_i(x_i) = x_i$.

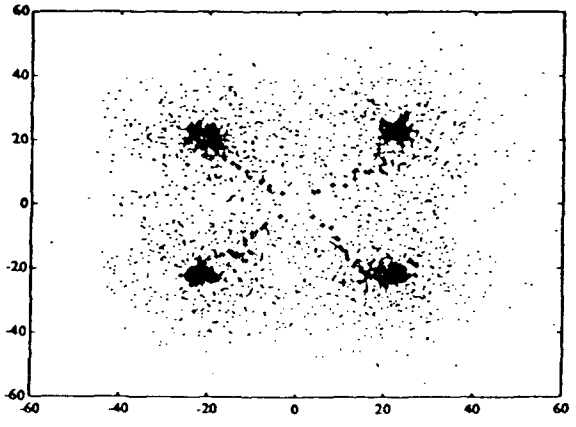


Fig. 5. Convergence of SCL1-trained synaptic vectors to Gaussian pattern-class centroids for the same pattern distribution as in Fig. 4.

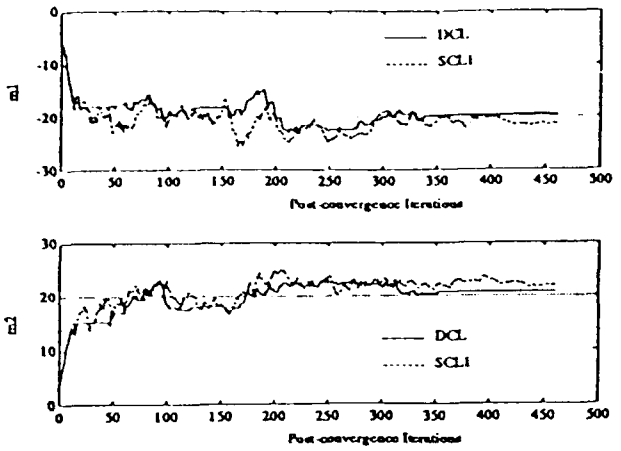


Fig. 6. Trajectories of the synaptic vectors after reaching the Gaussian pattern-class centroid at (−20, 20). Solid lines represent DCL-trained synaptic vectors. Dashed lines represent SCL1-trained synaptic vectors. The two graphs plot separately the m_1 and m_2 components of the synaptic vector $m = (m_1, m_2)$.

We calculated the mean-squared error (MSE) of centroid wandering for the class centered at (−20, 20) after 200 iterations. Other centroids produced comparable MSE of centroid wandering. In the first case, we used 540 Gaussian samples with variance 25. Then, for the DCL algorithm, the MSE of centroid wandering equaled 0.48. For the SCL1 algorithm, it equaled 1.48. In the second case, we used 554 Gaussian samples with variance 121. Then, for the DCL algorithm, the MSE of centroid wandering equaled 4. For the SCL1 algorithm, it equaled 7.11.

Next we compared the DCL system to the SCL1 and SCL2 systems for pattern classification accuracy. We trained each AVQ system with 500 Gaussian-distributed samples for each pattern class, and for each variance level centered about the same centroids (−20, 0) and (20, 0). We set variance levels at 20 units. For each variance level, we tested each AVQ system with 1000 new Gaussian-distributed samples for each pattern class. Fig. 7(a) shows the misclassification rates of the DCL, SCL1, and SCL2 systems for two representative Gaussian classes with equal variances. Fig. 7(b) shows misclassification performance for each AVQ system when we repeated the simulation in Fig. 7(a) for unequal variances. The pattern class with centroid (20, 0) had twice the variance of the pattern class with centroid (−20, 0). The three clustering algorithms behaved similarly for increasing variance values.

V. Phoneme Recognition Simulations

We obtained speech training samples from samples of continuous male speech with different English pronunciations. We used a time-dependent Fourier spectrum to extract features from the speech waveforms. An anti-alias low-pass filter prefiltered the speech signals. We then digitized the signals to 8 bits with a 10 kHz sampling frequency. A Hamming window divided the digitized speech signal into 256 sample segments. The fast Fourier transform algorithm gave 256 complex Fourier coefficients for each of the 256 windowed sample segments. We divided the 200 Hz–5 kHz frequency range into 16 regions. We divided the 200 Hz–3 kHz frequency range into 12 equal regions and the 3–5 kHz frequency range into four equal regions. Six Fourier coefficients represented each region in the 200 Hz–3 kHz range. 13 Fourier coefficients represented each region in the 3–5 kHz range. We calculated average power spectra over each region to form a 16-dimensional pattern vector. We produced 16-dimensional phoneme pattern vectors by repeatedly sliding the Hamming window by 100 samples.

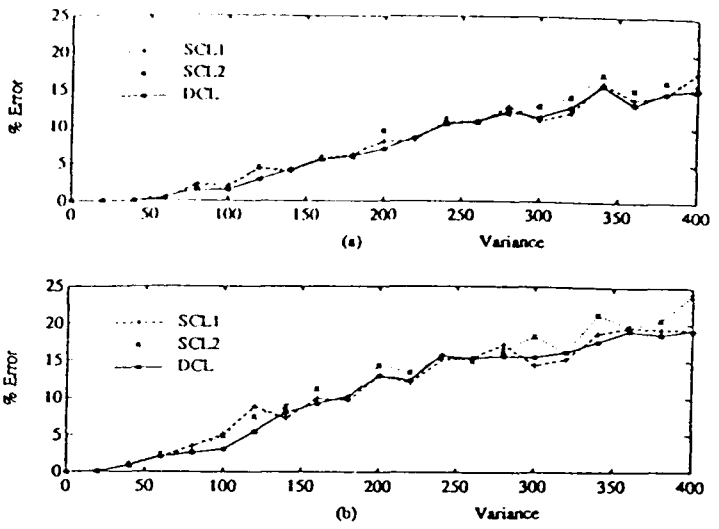


Fig. 7. AVQ misclassification rates for two Gaussian clusters: (a) with equal variance centered about the centroids (−20, 0) and (20, 0); and (b) with unequal variance. In (b), the pattern class centered about (20, 0) has twice the variance of the pattern class centered about (−20, 0).

The sample space consisted of real and artificial phonemes. The artificial phonemes were Gaussian random vectors with variation 9 centered at the real phoneme vectors. We generated these noisy phoneme samples to provide the AVQ systems with a statistically representative set of training samples.

The simulation compared the DCL, SCL1, and SCL2 learning systems for classification of nine representative English phonemes: five vowels /*a*, *e*, *i*, *o*, *u*/, two fricatives /*f*, *s*/, one nasal /*n*/, and one plosive sound /*t*/. Table I lists the misclassification rates. The AVQ systems tended to more accurately classify vowel and nasal sounds than they classified fricative and plosive sounds.

We trained each competitive AVQ system with 1000 Gaussian-distributed random phoneme vectors clustered into nine pattern classes. Each pattern class was centered about the original spoken phoneme and radially distributed with variance $\sigma^2 = 9$. We randomly selected training data according to a uniform probability distribution to simulate nine equiprobable pattern classes. We tested each AVQ system with 100 new Gaussian-distributed phoneme samples for each phoneme type. Except for the two phonemes /*o*/ and /*t*/, the DCL system misclassified no more frequently than the SCL systems.

	DCL	SCL1	SCL2
/a/	0	0	0
/e/	3	9	4
/i/	0	0	4
/o/	5	3	16
/u/	0	1	2
/f/	28	43	53
/s/	1	2	6
/n/	3	4	7
/t/	52	48	26

TABLE I
PERCENTAGE MISCLASSIFICATION RATES OF THE DCL, SCL1, AND SCL2 SYSTEMS FOR THE NINE ENGLISH PHONEMES */a, e, i, o, u, f, s, n, t/*

CONCLUSIONS

The DCL system performed well in centroid estimation and phoneme recognition. DCL synaptic vectors converged faster to centroids than did SCL1 synaptic vectors when logistic bipolar signal functions transformed the input sample. DCL synaptic vectors wandered less about pattern-class centroids than SCL synaptic vectors.

Our phoneme-recognition simulations were preliminary, but agreed with our centroid-estimation simulations. The phoneme-recognition simulations suggest that unsupervised DCL systems will perform as well as supervised SCL1 and SCL2 systems in many pattern environments, even though DCL systems use less pattern-class information.

In general, we do not know in advance whether $x \in D_i$ for every training sample x, and for every pattern class D_i, for an arbitrary classification, filtering, or estimation problem. We may not even know the approximate number or characteristics of the underlying decision classes. We can still apply DCL techniques in these cases and expect SCL-level performance. But we may never know how SCL systems would perform on the same data.

REFERENCES

[1] M. Gluck, D. Parker, and E. Reifsnider, "Some biological implications of a differential-Hebbian learning rule," *Psychobiol.*, vol. 16, no. 3, pp. 298-302, 1988.

[2] P. J. Huber, *Robust Statistics*. New York: Wiley, 1981.

[3] T. Kohonen, *Self-Organization and Associative Memory, 2nd edition*. New York: Springer-Verlag, 1988.

[4] ——, "The neural phonetic typewriter," *IEEE Comput. Mag.*, pp. 11-22, Mar. 1988.

[5] T. Kohonen, G. Barna, and R. Chrisley, "Statistical pattern recognition with neural networks: Benchmarking studies," in *Proc. Int. Conf. Neural Networks (ICNN-88)*, 1988, vol. I, pp. 61-68.

[6] B. Kosko, "Differential Hebbian learning," in *Proc. Amer. Inst. Phys. Conf.: Neural Networks Comput.*, Apr. 1986, pp. 277-282.

[7] ——, "Unsupervised learning in noise," *IEEE Trans. Neural Networks*, vol. 1, no. 1, pp. 44-57, Mar. 1990.

[8] ——, "Stochastic competitive learning," in *Proc. Summer 1990 Int. Joint Conf. Neural Networks (IJCNN-90)*, June 1990, vol. II, pp. 215-226.

[9] ——, *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*. Englewood Cliffs, NJ: Prentice-Hall, 1990.

[10] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Stat. Prob.*, 1967, pp. 281-297.

[11] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Stat.*, vol. 22, pp. 400-407, 1951.

[12] Y. Z. Tsypkin, *Foundations of the Theory of Learning Systems*. New York: Academic, 1973.

In: Kosko, B., NEURAL NETWORKS AND FUZZY SYSTEMS, Prentice-Hall, 1990

CHAPTER 17

FUZZY ASSOCIATIVE MEMORIES

Fuzzy Systems as Between-Cube Mappings

In Chapter 16, we introduced continuous or fuzzy sets as points in the unit hypercube $I^n = [0,1]^n$. Within the cube we were interested in the distance between points. This led to measures of the size and fuzziness of a fuzzy set and, more fundamentally, to a measure of how much one fuzzy set is a subset of another fuzzy set. This *within-cube* theory directly extends to the continuous case where the space X is a subset of R^n or, in general, where X is a subset of products of real or complex spaces.

The next step is to consider mappings *between* fuzzy cubes. This level of abstraction provides a surprising and fruitful alternative to the propositional and predicate-calculus reasoning techniques used in artificial-intelligence (AI) expert systems. It allows us to reason with sets instead of propositions.

The fuzzy set framework is numerical and multidimensional. The AI framework is symbolic and one-dimensional, with usually only bivalent expert "rules" or propositions allowed. Both frameworks can encode structured knowledge in linguistic form. But the fuzzy approach translates the structured knowledge into a flexible *numerical* framework and processes it in a manner that resembles neural network processing. The numerical framework also allows fuzzy systems to be adaptively inferred and modified, perhaps with neural or statistical techniques, directly from problem domain sample data.

Between-cube theory is fuzzy systems theory. A fuzzy set is a point in a cube. A fuzzy system is a mapping between cubes. A fuzzy system S maps fuzzy sets to fuzzy sets. Thus a **fuzzy system** S is a transformation $S : I^n \rightarrow I^p$. The n-dimensional unit hypercube I^n houses all the fuzzy subsets of the domain space, or input *universe of discourse*, $X = \{x_1, \ldots, x_n\}$. I^p houses all the fuzzy subsets of the range space, or output universe of discourse, $Y = \{y_1, \ldots, y_p\}$. X and Y can also be subsets of R^n and R^p. Then the fuzzy power sets $F(2^X)$ and $F(2^Y)$ replace I^n and I^p.

In general a fuzzy system S maps families of fuzzy sets to families of fuzzy sets, thus $S : I^{n_1} \times \ldots \times I^{n_r} \rightarrow I^{p_1} \times \ldots \times I^{p_s}$. Here too we can extend the definition of a fuzzy system to allow arbitrary products of arbitrary mathematical spaces to serve as the domain or range spaces of the fuzzy sets.

(A technical comment is in order for sake of historical clarification. A tenet, perhaps the defining tenet, of the classical theory [Dubois, 1980] of fuzzy sets as functions concerns the fuzzy extension of any mathematical function. This tenet holds that any function $f : X \rightarrow Y$ that maps points in X to points in Y can be extended to map the fuzzy subsets of X to the fuzzy subsets of Y. The so-called *extension principle* is used to define the set-function $f : F(2^X) \rightarrow F(2^Y)$, where $F(2^X)$ is the fuzzy power set of X, the set of all fuzzy subsets of X. The formal definition of the extension principle is complicated. The key idea is a supremum of pairwise minima. Unfortunately, the extension principle achieves generality at the price of triviality. One can show [Kosko, 1986a-87] that in general the extension principle extends functions to fuzzy sets by stripping the fuzzy sets of their fuzziness, mapping the fuzzy sets into bit vectors of nearly all 1s. This shortcoming, combined with the tendency of the extension-principle framework to push fuzzy theory into largely inaccessible regions of abstract mathematics, led in part to the development of the alternative sets-as-points geometric framework of fuzzy theory.)

We shall focus on fuzzy systems $S : I^n \rightarrow I^p$ that map *balls* of fuzzy sets in I^n to balls of fuzzy sets in I^p. These continuous fuzzy systems behave as associative memories. They map close inputs to close outputs. We shall refer to them as **fuzzy associative memories**, or FAMs.

The simplest FAM encodes the **FAM rule** or association (A_i, B_i), which associates

the p-dimensional fuzzy set B_i with the n-dimensional fuzzy set A_i. These minimal FAMs essentially map one ball in I^n to one ball in I^p. They are comparable to simple neural networks. But the minimal FAMs need not be adaptively trained. As discussed below, structured knowledge of the form "If traffic is heavy in this direction, then keep the stop light green longer" can be directly encoded in a Hebbian-style FAM matrix. In practice we can eliminate even this matrix. In its place the user encodes the fuzzy-set association (HEAVY, LONGER) as a single linguistic entry in a FAM bank matrix.

In general a **FAM system** $F : I^n \rightarrow I^p$ encodes and processes in parallel a **FAM bank** of m FAM rules $(A_1, B_1), \ldots, (A_m, B_m)$. Each input A to the FAM system activates each stored FAM rule to different degree. The minimal FAM that stores (A_i, B_i) maps input A to B_i', a partially activated version of B_i. The more A resembles A_i, the more B_i' resembles B_i. The corresponding output fuzzy set B combines these partially activated fuzzy sets $B_1', \ldots, B_m'$. In the simplest case B is a weighted average of the partially activated sets:

$$B = w_1 B_1' + \ldots + w_m B_m' ,$$

where w_i reflects the credibility, frequency, or strength of the fuzzy association (A_i, B_i). In practice we usually "defuzzify" the output waveform B to a single numerical value y_j in Y by computing the fuzzy centroid of B with respect to the output universe of discourse Y.

More general still, a FAM system encodes a bank of compound FAM rules that associate multiple output or consequent fuzzy sets $B_i^1, \ldots, B_i^s$ with multiple input or antecedent fuzzy sets $A_i^1, \ldots, A_i^r$. We can treat compound FAM rules as compound linguistic conditionals. Structured knowledge can then be naturally, and in many cases easily, obtained. We combine antecedent and consequent sets with logical conjunction, disjunction, or negation. For instance, we would interpret the compound association $(A^1, A^2;\ B)$ linguistically as the compound conditional "IF X^1 is A^1 AND X^2 is A^2 , THEN Y is B" if the comma in the fuzzy association $(A^1, A^2;\ B)$ stood for conjunction instead of, say, disjunction.

We specify in advance the numerical universes of discourse X^1, X^2, and Y. For each universe of discourse X, we specify an appropriate *library* of fuzzy set values, $A_1^r, \ldots, A_k^r$. Contiguous fuzzy sets in a library overlap. In principle a neural network can estimate these

libraries of fuzzy sets. In practice this is usually unnecessary. The library sets represent a weighted, though overlapping, quantization of the input space X. A different library of fuzzy sets similarly quantizes the output space Y. Once the library of fuzzy sets is defined, we construct the FAM by choosing appropriate combinations of input and output fuzzy sets. We can use adaptive techniques to make, assist, or modify these choices.

An **adaptive FAM** (AFAM) is a *time-varying* FAM system. System parameters gradually change as the FAM system samples and processes data. Below we discuss how neural network algorithms can adaptively infer FAM rules from training data. In principle learning can modify other FAM system components, such as the libraries of fuzzy sets or the FAM-rule weights w_i.

Below we propose and illustrate an unsupervised adaptive clustering scheme, based on competitive learning, for "blindly" generating and refining the bank of FAM rules. In some cases we can use supervised learning techniques, though we need additional information to accurately generate error estimates.

FUZZY AND NEURAL FUNCTION ESTIMATORS

Neural and fuzzy systems estimate sampled functions and behave as associative memories. They share a key advantage over traditional statistical-estimation and adaptive-control approaches to function estimation. They are *model-free* estimators. Neural and fuzzy systems estimate a function without requiring a mathematical description of how the output functionally depends on the input. They "learn from example." More precisely, they learn from samples.

Both approaches are numerical, can be partially described with theorems, and admit an algorithmic characterization that favors silicon and optical implementation. These properties distinguish neural and fuzzy approaches from the symbolic processing approaches of artificial intelligence.

Neural and fuzzy systems differ in how they estimate sampled functions. They differ in the kind of samples used, how they represent and store those samples, and how they

associatively "inference" or map inputs to outputs.

These differences appear during system construction. The neural approach requires the specification of a nonlinear dynamical system, usually feedforward, the acquisition of a sufficiently representative set of numerical training samples, and the encoding of those training samples in the dynamical system by repeated learning cycles. The fuzzy system requires only that a linguistic "rule matrix" be partially filled in. This task is markedly simpler than designing and training a neural network. Once we construct the systems, we can present the same numerical inputs to either system. The outputs will be in the same numerical space of alternatives. So both systems correspond to a surface or manifold in the input-output product space $X \times Y$. We present examples of these surfaces in Chapters 18 and 19.

Which system, neural or fuzzy, is more appropriate for a particular problem depends on the nature of the problem and the availability of numerical and structured data. To date fuzzy techniques have been most successfully applied to control problems. These problems often permit comparison with standard control-theoretic and expert-system approaches. Neural networks so far seem best applied to ill-defined two-class pattern recognition problems (defective or nondefective, bomb or not, etc.). The application of both approaches to new problem areas is just beginning, amid varying amounts of enthusiasm and scepticism.

Fuzzy systems estimate functions with *fuzzy set* samples (A_i, B_i). Neural systems use *numerical point* samples (x_i, y_i). Both kinds of samples are from the input-output product space $X \times Y$. Figure 17.1 illustrates the geometry of fuzzy-set and numerical-point samples taken from the function f: $X \rightarrow Y$.

The fuzzy-set association (A_i, B_i) is sometimes called a "rule." This is misleading since reasoning with sets is not the same as reasoning with propositions. Reasoning with sets is harder. Sets are multidimensional, and associations are housed in matrices, not conditionals. We must take care how we define each term and operation. We shall refer to the antecedent term A_i in the fuzzy association (A_i, B_i) as the **input associant** and the

consequent term B_i as the **output associant.**

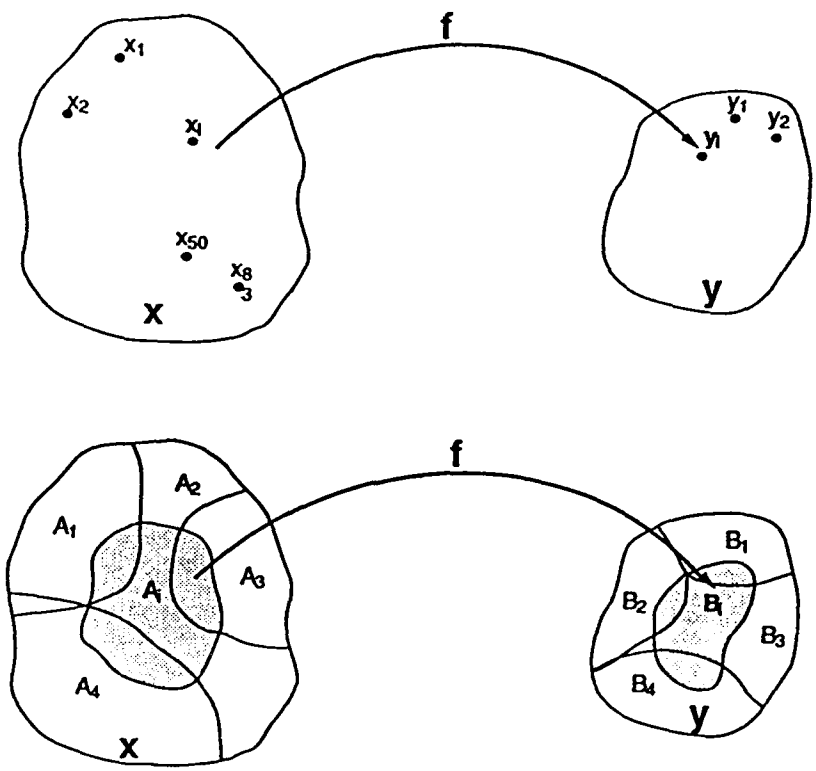


FIGURE 17.1 Function f maps domain X to range Y. In the first illustration we use several numerical point samples $(x_i,\ y_i)$ to estimate f: $X \longrightarrow Y$. In the second case we use only a few fuzzy subsets A_i of X and B_i of Y. The fuzzy association $(A_i,\ B_i)$ represents system structure, as an adaptive clustering algorithm might infer or as an expert might articulate. In practice there are

usually fewer different output associants or "rule" consequents B_i than input associants or antecedents A_i.

The fuzzy-set sample (A_i, B_i) encodes *structure*. It represents a mapping itself, a minimal *fuzzy association* of part of the output space with part of the input space. In practice this resembles a meta-rule—IF A_i, THEN B_i—the type of structured linguistic rule an expert might articulate to build an expert-system "knowledge base". The association might also be the result of an adaptive clustering algorithm.

Consider a fuzzy association that might be used in the intelligent control of a traffic light: "If the traffic is heavy in this direction, then keep the light green longer." The fuzzy association is (HEAVY, LONGER). Another fuzzy association might be (LIGHT, SHORTER). The fuzzy system encodes each linguistic association or "rule" in a numerical *fuzzy associative memory* (FAM) mapping. The FAM then numerically processes numerical input data. A measured description of traffic density (e.g., 150 cars per unit road surface area) then corresponds to a unique numerical output (e.g., 3 seconds), the "recalled" output.

The degree to which a particular measurement of traffic density is heavy depends on how we define the fuzzy set of heavy traffic. The definition may be obtained from statistical or neural clustering of historical data or from pooling the responses of experts. In practice the fuzzy engineer and the problem domain expert agree on one of many possible libraries of fuzzy set definitions for the variables in question.

The degree to which the traffic light is kept green longer depends on the degree to which the measurement is heavy. In the simplest case the two degrees are the same. In general they differ. In actual fuzzy systems the output control variables—in this case the single variable green light duration—depend on many FAM rule antecedents or associants that are activated to different degrees by incoming data.

Neural vs. Fuzzy Representation of Structured Knowledge

The functional distinction between how fuzzy and neural systems differ begins with how they represent structured knowledge. How would a neural network encode the same associative information? How would a neural network encode the structured knowledge "If the traffic is heavy in this direction, then keep the light green longer"?

The simplest method is to encode two associated numerical vectors. One vector represents the input associant HEAVY. The other vector represents the output associant LONGER. But this is too simple. For the neural network's fault tolerance now works to its disadvantage. The network tends to reconstruct partial inputs to complete sample inputs. It erases the desired partial degrees of activation. If an input is close to A_i, the output will tend to be B_i. If the output is distant from A_i, the output will tend to be some other sampled output vector or a spurious output altogether.

A better neural approach is to encode a mapping from the heavy-traffic subspace to the longer-time subspace. Then the neural network needs a representative sample set to capture this structure. Statistical networks, such as adaptive vector quantizers, may need thousands of statistically representative samples. Feedforward multi-layer neural networks trained with the backpropagation algorithm may need hundreds of representative numerical input-output pairs and may need to recycle these samples tens of thousands of times in the learning process.

The neural approach suffers a deeper problem than just the computational burden of training. *What* does it encode? How do we know the network encodes the original structure? What does it recall? There is no natural inferential audit trail. System nonlinearities wash it away. Unlike an expert system, we do not know which inferential paths the network uses to reach a given output or even which inferential paths exist. There is only a system of synchronous or asynchronous nonlinear functions. Unlike, say, the adaptive Kalman filter, we cannot appeal to a postulated mathematical model of how the output state depends on the input state. Model-free estimation is, after all, the central computational advantage of neural networks. The cost is system inscrutability.

We are left with an unstructured computational black box. We do not know what the neural network encoded during training or what it will encode or forget in further training. (For competitive adaptive vector quantizers we do know that sample-space centroids are asymptotically estimated.) We can characterize the neural network's behavior only by exhaustively passing all inputs through the black box and recording the recalled outputs. The characterization may be in terms of a summary scalar like mean-squared error.

This black-box characterization of the network's behavior involves a computational *dilemma*. On the one hand, for most problems the number of input-output cases we need to check is computationally prohibitive. On the other, when the number of input-output cases is tractable, we may as well store these pairs and appeal to them directly, and without error, as a look-up table. In the first case the neural network is unreliable. In the second case it is unnecessary.

A further problem is sample generation. Where did the original numerical point samples come from? Was an expert asked to give numbers? How reliable are such numerical vectors, especially when the expert feels most comfortable giving the original linguistic data? This procedure seems at most as reliable as the expert-system method of asking an expert to give condition-action rules with numerical uncertainty weights.

Statistical neural estimators require a "statistically representative" sample set. We may need to randomly "create" these samples from an initial small sample set by bootstrap techniques or by random-number generation of points clustered near the original samples. Both sample-augmentation procedures assume that the initial sample set sufficiently represents the underlying probability distribution. The problem of where the original sample set comes from remains. The fuzziness of the notion "statistically representative" compounds the problem. In general we do not know in advance how well a given sample set reflects an unknown underlying distribution of points. Indeed when the network is adapting on-line, we know only past samples. The remainder of the sample set is in the unsampled future.

In contrast, fuzzy systems directly encode the linguistic sample (HEAVY, LONGER) in a dedicated numerical matrix. The default encoding technique is the fuzzy Hebb procedure discussed below. For practical problems, as mentioned above, the numerical matrix need not be stored. Indeed it need not even be formed. Certain numerical inputs permit this

simplification, as we shall see below. In general we describe inputs by an uncertainty distribution, probabilistic or fuzzy. Then we must use the entire matrix.

For instance, if a *heavy traffic* input is simply the number 150, we can omit the FAM matrix. But if the input is a Gaussian curve with mean 150, then in principle we must process the vector input with a FAM matrix. (In practice we might use only the mean.) This difference is explained below. The dimensions of the linguistic FAM bank matrix are usually small. The dimensions reflect the quantization levels of the input and output spaces.

The fuzzy approach combines the purely numerical approaches of neural networks and mathematical modeling with the symbolic, structure-rich approaches of artificial intelligence. We acquire knowledge symbolically—or numerically if we use adaptive techniques —but represent it numerically. We also process data numerically. Adaptive FAM rules correspond to common-sense, often non-articulated, behavioral rules that improve with experience.

We can acquire structured expertise in the fuzzy terminology of the knowledge source, the "expert." This requires little or no force-fitting. Such is the expressive power of fuzziness. Yet in the numerical domain we can prove theorems and design hardware.

This approach does not abandon neural network techniques. Instead, it limits them to *unstructured* parameter and state estimation, pattern recognition, and cluster formation. The system *architecture* remains fuzzy, though perhaps adaptively so. In the same spirit, no one believes that the brain is a single unstructured neural network.

FAMS as Mappings

Fuzzy associative memories (FAMs) are transformations. *FAMs map fuzzy sets to fuzzy sets.* They map unit cubes to unit cubes. This is evident in Figure 17.1. In the simplest case the FAM consists of a single association, such as (HEAVY, LONGER). In general the FAM consists of a bank of different FAM associations. Each association is represented by a different numerical FAM matrix, or a different entry in a FAM-bank

matrix. These matrices are not combined as with neural network associative memory (outer-product) matrices. (An exception is the *fuzzy cognitive map* [Kosko, 1988; Taber, 1987, 1990].) The matrices are stored separately but accessed in parallel.

We begin with single-association FAMs. For concreteness let the fuzzy-set pair (A, B) encode the traffic-control association (HEAVY, LIGHT). We quantize the domain of traffic density to the n numerical variables $x_1, x_2, \ldots, x_n$. We quantize the range of green-light duration to the p variables $y_1, y_2, \ldots, y_p$. The elements x_i and y_j belong respectively to the ground sets $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_p\}$. x_1 might represent zero traffic density. y_p might represent 10 seconds.

The fuzzy sets A and B are fuzzy subsets of X and Y. So A is point in the n-dimensional unit hypercube $I^n = [0, 1]^n$, and B is a point in the p-dimensional fuzzy cube I^p. Equivalently, we can think of A and B as membership functions m_A and m_B mapping the elements x_i of X and y_j of Y to degrees of membership in $[0, 1]$. The membership values, or *fit* (fuzzy unit) values, indicate how much x_i belongs to or fits in subset A, and how much y_j belongs to B. We describe this with the abstract functions $m_A : X \longrightarrow [0, 1]$ and $m_B : Y \longrightarrow [0, 1]$. We shall freely view sets both as functions and as points.

The geometric *sets-as-points* interpretation of fuzzy sets A and B as points in unit cubes allows a natural vector representation. We represent A and B by the numerical *fit vectors* $A = (a_1, \ldots, a_n)$ and $B = (b_1, \ldots, b_p)$, where $a_i = m_A(x_i)$ and $b_j = m_B(y_j)$. We can interpret the identifications A = HEAVY and B = LONGER to suit the problem at hand. Intuitively the a_i values should increase as the index i increases, perhaps approximating a sigmoid membership function. Figure 17.2 illustrates three possible fuzzy subsets of the universe of discourse X.

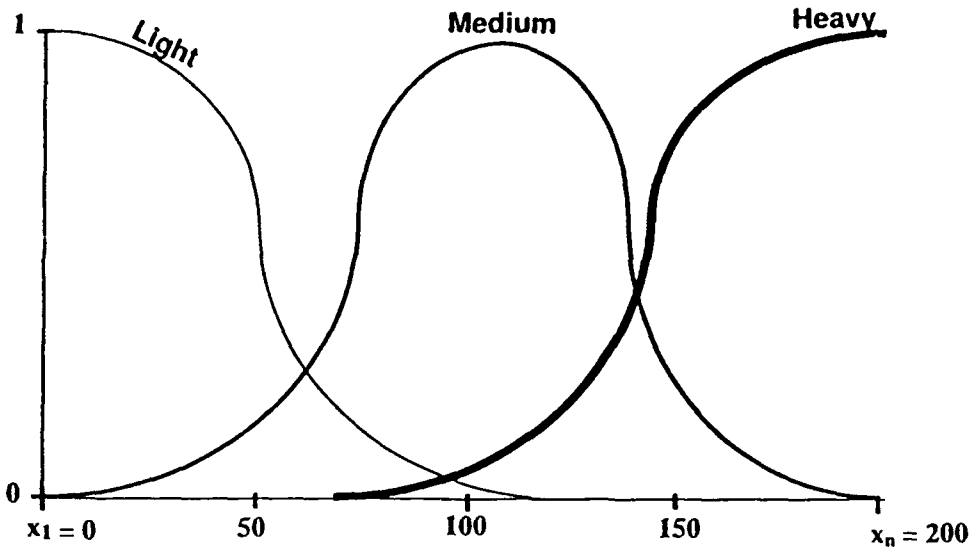


FIGURE 17.2 Three possible fuzzy subsets of traffic density space X. Each fuzzy sample corresponds to such a subset. We draw the fuzzy sets as continuous membership functions. In practice membership values are quantized. So the sets are points in the unit hypercube I^n. Each fuzzy sample corresponds to such a subset.

Fuzzy Vector-Matrix Multiplication: Max-Min Composition

Fuzzy vector-matrix multiplication is similar to classical vector-matrix multiplication. We replace pairwise multiplications with pairwise minima. We replace column (row) sums with column (row) maxima. We denote this **fuzzy vector-matrix composition** relation, or the **max-min composition** relation [Klir, 1988], by the composition operator "o". For row fit vectors A and B and fuzzy n-by-p matrix M (a point in $I^{n \times p}$):

$$A \circ M = B \quad , \tag{1}$$

where we compute the "recalled" component b_j by taking the fuzzy inner product of fit vector A with the jth column of M:

$$b_j = \max_{1 \le i \le n} \min(a_i, m_{ij}) \quad . \tag{2}$$

Suppose we compose the fit vector $A = (.3\ .4\ .8\ 1)$ with the fuzzy matrix M given by

$$M = \begin{pmatrix} .2 & .8 & .7 \\ .7 & .6 & .6 \\ .8 & .1 & .5 \\ 0 & .2 & .3 \end{pmatrix} .$$

Then we compute the "recalled" fit vector $B = A \circ M$ component-wise as

$$\begin{aligned} b_1 &= \max\{\min(.3,\ .2),\ \min(.4,\ .7),\ \min(.8,\ .8),\ \min(1,\ 0)\} \\ &= \max(.2,\ .4,\ .8,\ 0) \\ &= .8\ , \\ b_2 &= \max(.3,\ .4,\ .1,\ .2) \\ &= .4\ , \\ b_3 &= \max(.3,\ .4,\ .5,\ .3) \\ &= .5\ . \end{aligned}$$

So $B = (.8\ .4\ .5)$. If we somehow encoded $(A,\ B)$ in the FAM matrix M, we would say that the FAM system exhibits *perfect recall* in the forward direction.

The neural interpretation of max-min composition is that each neuron in field F_Y (or field F_B) generates its signal/activation value by fuzzy linear composition. Passing

information back through M^T allows us to interpret the fuzzy system as a bidirectional associative memory (BAM). The Bidirectional FAM Theorems below characterize successful BAM recall for fuzzy correlation or Hebbian learning.

For completeness we also mention the **max-product composition** operator, which replaces minimum with product in (2):

$$b_j = \max_{1 \le i \le n} a_i \, m_{ij} \quad .$$

In the fuzzy literature this composition operator is often confused with the fuzzy correlation encoding scheme discussed below. Max-product composition is a method for "multiplying" fuzzy matrices or vectors. Fuzzy correlation, which also uses pairwise products of fit values, is a method for constructing fuzzy matrices. In practice, and in the following discussion, we use only max-min composition.

FUZZY HEBB FAMs

Most fuzzy systems found in applications are fuzzy Hebb FAMs [Kosko, 1986b]. They are fuzzy systems $S : I^n \rightarrow I^p$ constructed in a simple neural-like manner. As discussed in Chapter 4, in neural network theory we interpret the classical Hebbian hypothesis of correlation synaptic learning [Hebb, 1949] as unsupervised learning with the signal product $S_i \, S_j$:

$$\dot{m}_{ij} = -m_{ij} + S_i(x_i) \, S_j(y_j) \quad . \tag{3}$$

For a given pair of bipolar vectors (X, Y), the neural interpretation gives the *outer-product* correlation matrix

$$M = X^T Y \quad . \tag{4}$$

The **fuzzy Hebb matrix** is similarly defined pointwise by the minimum of the "signals" a_i and b_j, an encoding scheme we shall call **correlation-minimum encoding**:

$$m_{ij} \;=\; \min(a_i, b_j) \quad , \tag{5}$$

given in matrix notation as the *fuzzy outer-product*

$$M \;=\; A^T \circ B \quad . \tag{6}$$

Mamdani [1977] and Togai [1986] independently arrived at the fuzzy Hebbian prescription (5) as a multi-valued logical-implication operator: $\text{truth}(a_i \rightarrow b_i) = \min(a_i, b_j)$. The min operator, though, is a symmetric truth operator. So it does not properly generalize the classical implication $P \rightarrow Q$, which is false if and only if the antecedent P is true and the consequent Q is false, $t(P) = 1$ and $t(Q) = 0$. In contrast, a like desire to define a "conditional possibility" matrix pointwise with continuous implication values led Zadeh [1983] to choose the Lukasiewicz implication operator: $m_{ij} = \text{truth}(a_i \rightarrow b_j) = \min(1, 1 - a_i + b_j)$. The problem with the Lukasiewicz operator is that it usually unity. For $\min(1, 1 - a_i + b_j) < 1$ iff $a_i > b_j$. Most entries of the resulting matrix M are unity or near unity. This ignores the information in the association (A, B). So $A' \circ M$ tends to equal the largest fit value a'_k for any system input A'.

We construct an ***autoassociative*** fuzzy Hebb FAM matrix by encoding the redundant pair (A, A) in (6), as the fuzzy auto-correlation matrix:

$$M \;= A^T \circ A \quad . \tag{7}$$

In the previous example the matrix M was such that the input $A = (.3 \;\; .4 \;\; .8 \;\; 1)$ recalled fit vector $B = (.8 \;\; .4 \;\; .5)$ upon max-min composition: $A \circ M = B$. Will B still be recalled if we replace the original matrix M with the fuzzy Hebb matrix found with (6)? Substituting A and B in (6) gives

$$M \;=\; A^T \circ B \;=\; \begin{pmatrix} .3 \\ .4 \\ .8 \\ 1 \end{pmatrix} \circ (.8 \;\; .4 \;\; .5) \;=\; \begin{pmatrix} .3 & .3 & .3 \\ .4 & .4 & .4 \\ .8 & .4 & .5 \\ .8 & .4 & .5 \end{pmatrix} \quad .$$

This fuzzy Hebb matrix M' illustrates two key properties. First, the ith row of M is the pairwise minimum of a_i and the output associant B. Symmetrically, the jth column of M is the pairwise minimum of b_j and the input associant A:

$$M = \begin{bmatrix} a_1 \wedge B \\ \hline \vdots \\ \hline a_n \wedge B \end{bmatrix} \tag{8}$$

$$= [b_1 \wedge A^T \mid \ldots \mid b_m \wedge A^T] \quad , \tag{9}$$

where the cap operator denotes pairwise minimum: $a_i \wedge b_j = \min(a_i, b_j)$. The term $a_i \wedge B$ indicates component-wise minimum:

$$a_i \wedge B = (a_i \wedge b_1, \ldots, a_i \wedge b_n) \quad . \tag{10}$$

Hence if some $a_k = 1$, then the kth row of M is B. If some $b_l = 1$, the lth column of M is A. More generally, if some a_k is at least as large as every b_j, then the kth row of the fuzzy Hebb matrix M is B.

Second, the third and fourth ~~columns~~ rows of M are just the fit vector B. Yet no column is A. This allows perfect recall in the forward direction, $A \circ M = B$, but not in the backward direction, $B \circ M^T \neq A$:

$$A \circ M = (.8 \ .4 \ .5) = B \quad ,$$

$$B \circ M^T = (.3 \ .4 \ .8 \ .8) = A' \subset A \quad .$$

A' is a proper subset of A: $A' \neq A$ and $S(A', A) = 1$, where S measures the degree of subsethood of A' in A, as discussed in Chapter 16. In other words, $a'_i \leq a_i$ for each i and $a'_k < a_k$ for at least one k. The Bidirectional FAM Theorems below show that this is a general property: If $B' = A \circ M$ differs from B, then B' is a proper subset of B. Hence fuzzy subsets truly map to fuzzy subsets.

The Bidirectional FAM Theorem for Correlation-Minimum Encoding

Analysis of FAM recall uses the traditional [Klir, 1988] fuzzy set notions of the *height* and the *normality* of fuzzy sets. The **height** $H(A)$ of fuzzy set A is the maximum fit value of A:

$$H(A) = \max_{1 \le i \le n} a_i \quad .$$

A fuzzy set is **normal** if $H(A) = 1$, if at least one fit value a_k is maximal: $a_k = 1$. In practice fuzzy sets are usually normal. We can extend a nonnormal fuzzy set to a normal fuzzy set by adding a dummy dimension with corresponding fit value $a_{n+1} = 1$.

Recall accuracy in fuzzy Hebb FAMs constructed with correlation-minimum encoding depends on the heights $H(A)$ and $H(B)$. Normal fuzzy sets exhibit perfect recall. Indeed (A, B) is a bidirectional fixed point—$A \circ M = B$ and $B \circ M^T = A$—if and only if $H(A) = H(B)$, which always holds if A and B are normal. This is the content of the Bidirectional FAM Theorem [Kosko, 1986a] for correlation-minimum encoding. Below we present a similar theorem for correlation-product encoding.

Correlation-Minimum Bidirectional FAM Theorem. If $M = A^T \circ B$, then

(i) $A \circ M = B$ iff $H(A) \ge H(B)$,

(ii) $B \circ M^T = A$ iff $H(B) \ge H(A)$,

(iii) $A' \circ M \subset B$ for any A' .

(iv) $B' \circ M^T \subset A$ for any B' .

Proof. Observe that the height $H(A)$ is the *fuzzy norm* of A:

$$A \circ A^T = \max_i a_i \wedge a_i = \max_i a_i = H(A) \ .$$

Then

$$\begin{aligned} A \circ M &= A \circ (A^T \circ B) \\ &= (A \circ A^T) \circ B \\ &= H(A) \circ B \\ &= H(A) \wedge B \ . \end{aligned}$$

So $H(A) \wedge B = B$ iff $H(A) \geq H(B)$, establishing (i). Now suppose A' is an arbitrary fit vector in I^n. Then

$$\begin{aligned} A' \circ M &= (A' \circ A^T) \circ B \\ &= (A' \circ A^T) \wedge B \ , \end{aligned}$$

which establishes (iii). A similar argument using $M^T = B^T \circ A$ establishes (ii) and (iv). Q.E.D.

The equality $A \circ A^T = H(A)$ implies an immediate corollary of the Bidirectional FAM Theorem. Supersets $A' \supset A$ behave the same as the encoded input associant A: $A' \circ M = B$ if $A \circ M = B$. Fuzzy Hebb FAMs ignore the information in the difference $A' - A$, when $A' \subset A'$.

Correlation-Product Encoding

An alternative fuzzy Hebbian encoding scheme is **correlation-product encoding**. The standard mathematical outer product of the fit vectors A and B forms the FAM matrix M. This is given pointwise as

$$m_{ij} \;=\; a_i \, b_j \quad , \tag{11}$$

and in matrix notation as

$$M \;=\; A^T \, B \quad . \tag{12}$$

So the ith row of M is just the fit-scaled fuzzy set $a_i \, B$, and the jth column of M is $b_j \, A^T$:

$$M \;=\; \left[\begin{array}{c} a_1 \, B \\ \hline \vdots \\ \hline a_n \, B \end{array}\right] \tag{13}$$

$$\;=\; [b_1 \, A^T \mid \ldots \mid b_m \, A^T] \quad , \tag{14}$$

If $A \;=\; (.3 \;\; .4 \;\; .8 \;\; 1)$ and $B \;=\; (.8 \;\; .4 \;\; .5)$ as above, we encode the FAM rule $(A, \; B)$ with correlation-product in the following matrix M:

$$M \;=\; \begin{pmatrix} .24 & .12 & .15 \\ .32 & .16 & .2 \\ .64 & .32 & .4 \\ .8 & .4 & .5 \end{pmatrix} \quad .$$

Note that if $A' \;=\; (0 \; 0 \; 0 \; 1)$, then $A' \circ M \;=\; B$. The output associant B is recalled to maximal degree. If $A' \;=\; (1 \; 0 \; 0 \; 0)$, then $A' \circ M \;=\; (.24 \;\; .12 \;\; .15)$. The output B is recalled only to degree .3.

Correlation-minimum encoding produces a matrix of clipped B sets. Correlation-product encoding produces a matrix of scaled B sets. In membership function plots, the scaled fuzzy sets $a_i \, B$ all have the same shape as B. The clipped fuzzy sets $a_i \wedge B$ are largely flat. In this sense correlation-product encoding preserves more information than correlation-minimum encoding, an important point in fuzzy applications when output fuzzy sets are added together as in equation (17) below. In the fuzzy-applications literature this often leads to the selection of correlation-product encoding.

Unfortunately, in the fuzzy-applications literature the correlation-product *encoding* scheme is invariably confused with the max-product composition method of recall or *inference*, as mentioned above. This confusion is so widespread it warrants formal clarification.

In practice, and in the fuzzy control applications developed in Chapters 18 and 19, the input fuzzy set A' is a binary vector with one 1 and all other elements 0—a row of the n-by-n identity matrix. A' represents the occurrence of the crisp measurement datum x_i, such as a traffic density value of 30. When applied to the encoded FAM rule (A, B), the measurement value x_i activates A to degree a_i. This is part of the max-min composition recall process, for $A' \circ M = (A' \circ A^T) \circ B = a_i \wedge B$ or $a_i B$ depending on whether correlation-minimum or correlation-product encoding is used. We activate or "fire" the output associant B of the "rule" to degree a_i.

Since the values a_i are binary, $a_i \, m_{ij} = a_i \wedge m_{ij}$. So the max-min and max-product composition operators coincide. We avoid this confusion by referring to both the recall process and the correlation encoding scheme as **correlation-minimum inference** when correlation-minimum encoding is combined with max-min composition, and as **correlation-product inference** when correlation-product encoding is combined with max-min composition.

We now prove the correlation-product version of the Bidirectional FAM Theorem.

Correlation-Product Bidirectional FAM Theorem. If $M = A^T B$ and A and B are non-null fit vectors, then

(i) $A \circ M = B$ iff $H(A) = 1$,

(ii) $B \circ M^T = A$ iff $H(B) = 1$,

(iii) $A' \circ M \subset B$ for any A' .

(iv) $B' \circ M^T \subset A$ for any B' .

Proof.

$$\begin{aligned} A \circ M &= A \circ (A^T B) \\ &= (A \circ A^T) B \\ &= H(A) B \ . \end{aligned}$$

Since B is not the empty set, $H(A) B = B$ iff $H(A) = 1$, establishing (i). ($A \circ M = B$ holds trivially if B is the empty set.) For an arbitrary fit vector A' in I^n:

$$\begin{aligned} A' \circ M &= (A' \circ A^T) B \\ &\subset H(A) B \\ &\subset B \ , \end{aligned}$$

since $A' \circ A \leq H(A)$, establishing (iii). (ii) and (iv) are proved similarly using $M^T = B^T A$. Q.E.D.

Superimposing FAM Rules

Now suppose we have m FAM rules or associations $(A_1, B_1), \ldots, (A_m, B_m)$. The fuzzy Hebb encoding scheme (6) leads to m FAM matrices $M_1, \ldots, M_m$ to encode the associations. The natural neural-network temptation is to add, or in this case maximum, the m matrices pointwise to distributively encode the associations in a single matrix M:

$$M = \max_{1 \leq k \leq m} M_k \ . \tag{15}$$

This superimposition scheme fails for fuzzy Hebbian encoding. The superimposed result tends to be the matrix $A^T \circ B$, where A and B are the pointwise maximum of the respective m fit vectors A_k and B_k. We can see this from the pointwise inequality

$$\max_{1 \leq k \leq m} \min(a_i^k, b_j^k) \leq \min(\max_{1 \leq k \leq m} a_i^k, \max_{1 \leq k \leq m} b_j^k) \quad . \tag{16}$$

Inequality (16) tends to hold with equality as m increases since all maximum terms approach unity. We lose the information in the m associations (A_k, B_k).

The fuzzy approach to the superimposition problem is to *additively superimpose the m recalled vectors* B_k' instead of the fuzzy Hebb matrices M_k. B_k' and M_k are given by

$$\begin{aligned} A \circ M_k &= A \circ (A_k^T \circ B_k) \\ &= B_k' \quad , \end{aligned}$$

for any fit-vector input A applied in parallel to the bank of FAM rules (A_k, B_k). This requires separately storing the m associations (A_k, B_k), as if each association in the FAM bank were a separate feedforward neural network.

Separate storage of FAM associations is costly but provides an "audit trail" of the FAM inference procedure. The user can directly determine which FAM rules contributed how much membership activation to a "concluded" output. Separate storage also provides knowledge-base modularity. The user can add or delete FAM-structured knowledge without disturbing stored knowledge. Both of these benefits are advantages over a pure neural-network architecture for encoding the same associations (A_k, B_k). Of course we can use neural networks exogenously to estimate, or even individually house, the associations (A_k, B_k).

Separate storage of FAM rules brings out another distinction between FAM systems and neural networks. A fit-vector input A activates all the FAM rules (A_k, B_k) in parallel but to different degrees. If A only partially "satisfies" the antecedent associant A_k, the consequent associant B_k is only partially activated. If A does not satisfy A_k at all, B_k does not activate at all. B_k' is the null vector.

Neural networks behave differently. They try to reconstruct the entire association (A_k, B_k) when stimulated with A. If A and A_k mismatch severely, a neural network will

tend to emit a non-null output B'_k, perhaps the result of the network dynamical system falling into a "spurious" attractor in the state space. This may be desirable for metrical classification problems. It is undesirable for inferential problems and, arguably, for associative memory problems. When we ask an expert a question outside his field of knowledge, in many cases it is more prudent for him to give no response than to give an educated, though wild, guess.

Recalled Outputs and "Defuzzification"

The recalled fit-vector output B is a weighted sum of the individual recalled vectors B'_k:

$$B = \sum_{k=1}^{m} w_k B'_k \quad , \tag{17}$$

where the nonnegative weight w_k summarizes the credibility or strength of the kth FAM rule (A_k, B_k). The credibility weights w_k are immediate candidates for adaptive modification. In practice we choose $w_1 = \ldots = w_m = 1$ as a default.

In principle, though not in practice, the recalled fit-vector output is a normalized sum of the B'_k fit vectors. This keeps the components of B unit-interval valued. We do not use normalization in practice because we invariably "defuzzify" the output distribution B to produce a single numerical output, a single value in the output universe of discourse $Y = \{y_1, \ldots, y_p\}$. The information in the output waveform B resides largely in the relative values of the membership degrees.

The simplest defuzzification scheme is to choose that element $y_{\max}$ that has maximal membership in the output fuzzy set B:

$$m_B(y_{\max}) = \max_{1 \le j \le k} m_B(y_j) \quad . \tag{18}$$

The popular probabilistic methods of maximum-likelihood and maximum-a-posteriori parameter estimation motivate this **maximum-membership defuzzification** scheme. The

maximum-membership scheme (18) is also computationally light.

There are two fundamental problems with the maximum-membership defuzzification scheme. First, the mode of the B distribution is not unique. This is especially troublesome with correlation-minimum encoding, as the representation (8) shows, and somewhat less troublesome with correlation-product encoding. Since the minimum operator clips off the top of the B_k fit vectors, the additively combined output fit vector B tends to be flat over many regions of universe of discourse Y. For continuous membership functions this leads to infinitely many modes. Even for quantized fuzzy sets, there may be many modes.

In practice we can average multiple modes. For large FAM banks of "independent" FAM rules, some form of the Central Limit Theorem (whose proof ultimately depends on Fourier transformability not probability) tends to apply. The waveform B tends to resemble a Gaussian membership function. So a unique mode tends to emerge. It tends to emerge with fewer samples if we use correlation-product encoding.

Second, the maximum-membership scheme ignores the information in much of the waveform B. Again correlation-minimum encoding compounds the problem. In practice B is often highly asymmetric, even if it is unimodal. Infinitely many output distributions can share the same mode.

The natural alternative is the **fuzzy centroid defuzzification** scheme. We directly compute the real-valued output as a normalized convex combination of fit values, the *fuzzy centroid* $\bar{B}$ of fit-vector B with respect to output space Y:

$$\bar{B} = \frac{\sum_{j=1}^{p} y_j \, m_B(y_j)}{\sum_{j=1}^{p} m_B(y_j)} \quad . \tag{19}$$

The fuzzy centroid is unique and uses all the information in the output distribution B. For symmetric unimodal distributions the mode and fuzzy centroid coincide. In many cases we must replace the discrete sums in (19) with integrals over continuously infinite spaces. We show in Chapter 19, though, that for libraries of trapezoidal fuzzy sets we can replace such a ratio of integrals with a ratio of simple discrete sums.

Note that computing the centroid (19) is the only step in the FAM inference procedure

that requires division. All other operations are inner products, pairwise minima, and additions. This promises realization in a fuzzy optical processor. Already some form of this FAM-inference scheme has led to digital [Togai, 1986] and analog [Yamakawa, 1987-88] VLSI circuitry.

FAM System Architecture

Figure 17.3 schematizes the architecture of the nonlinear FAM system F. Note that F maps fuzzy sets to fuzzy sets: $F(A) = B$. So F is in fact a fuzzy-system transformation $F : I^n \rightarrow I^p$. In practice A is a bit vector with one unity value, $a_i = 1$, and all other fit values zero, $a_j = 0$.

The output fuzzy set B is usually defuzzified with the centroid technique to produce an exact element y_j in the output universe of discourse Y. In effect defuzzification produces an output binary vector O, again with one element 1 and the rest 0s. At this level the FAM system F maps sets to sets, reducing the fuzzy system F to a mapping between Boolean cubes, $F : \{0,1\}^n \rightarrow \{0,1\}^p$. In many applications we model X and Y as continuous universes of discourse. So n and p are quite large. We shall call such systems **binary input-output FAMs**.

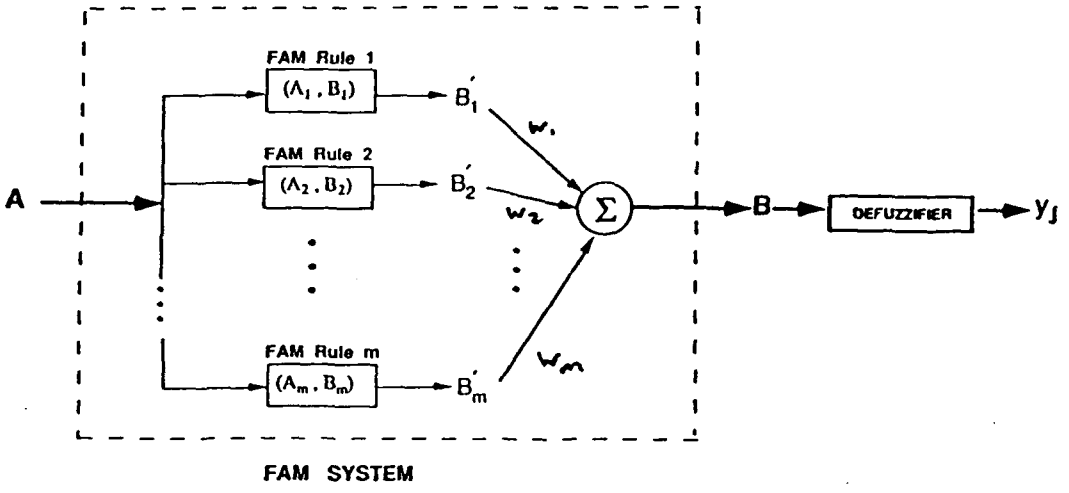


FIGURE 17.3 FAM system architecture. The FAM system F maps fuzzy sets in the unit cube I^n to fuzzy sets in the unit cube I^p. Binary input fuzzy sets are often used in practice to model exact input data. In general only an uncertainty estimate of the system state is available. So A is a proper fuzzy set. The user can defuzzify output fuzzy set B to yield exact output data, reducing the FAM system to a mapping between Boolean cubes.

Binary Input-Output FAMs: Inverted Pendulum Example

Binary input-output FAMs (BIOFAMs) are the most popular fuzzy systems for applications. BIOFAMs map system state-variable data to control data. In the case of traffic control, a BIOFAM maps traffic densities to green (and red) light durations.

BIOFAMs easily extend to multiple FAM rule antecedents, to mappings from product cubes to product cubes. There has been little theoretical justification for this extension,

aside from Mamdani's [1977] original suggestion to multiply relational matrices. The extension to multi-antecedent FAM rules is easier applied than formally explained. In the next section we present a general explanation for dealing with multi-antecedent FAM rules. First, though, we present the BIOFAM algorithm by illustrating it, and the FAM construction procedure, on an archetypical control problem.

Consider an inverted pendulum. In particular, consider how to adjust a motor to balance an inverted pendulum in two dimensions. The inverted pendulum is a classical control problem. It admits a math-model control solution. This provides a formal benchmark for BIOFAM pendulum controllers.

There are two state variables and one control variable. The first state variable is the *angle* θ that the pendulum shaft makes with the vertical. Zero angle corresponds to the vertical position. Positive angles are to the right of the vertical, negative angles to the left.

The second state variable is the *angular velocity* $\Delta\theta$. In practice we approximate the instantaneous angular velocity $\Delta\theta$ as the difference between the present angle measurement θ_t and the previous angle measurement θ_{t-1}:

$$\Delta\theta_t = \theta_t - \theta_{t-1} \quad .$$

The control variable is the motor current or *angular velocity* v_t. The velocity can also be positive or negative. We expect that if the pendulum falls to the right, the motor velocity should be negative to compensate. If the pendulum falls to the left, the motor velocity should be positive. If the pendulum successfully balances at the vertical, the motor velocity should be zero.

The real line R is the universe of discourse of the three variables. In practice we restrict each universe of discourse to a comparatively small interval, such as $[-90, 90]$ for the pendulum angle, centered about zero.

We can quantize each universe of discourse into five overlapping fuzzy sets. We know that the system variables can be positive, zero, or negative. We can quantize the magnitudes of the system variables finely or coarsely. Suppose we quantize the magnitudes as small, medium, and large. This leads to seven linguistic fuzzy set values:

NL: Negative Large

NM: Negative Medium

NS: Negative Small

ZE: Zero

PS: Positive Small

PM: Positive Medium

PL: Positive Large

For example, θ is a fuzzy *variable* that takes NL as a fuzzy set *value*. Different fuzzy quantizations of the angle universe of discourse allow the fuzzy variable θ to assume different fuzzy set values. The expressive power of the FAM approach stems from these fuzzy-set quantizations. In one stroke we reduce system dimensions, and we describe a nonlinear numerical process with linguistic common-sense terms.

We are not concerned with the exact shape of the fuzzy sets defined on each of the three universes of discourse. In practice the quantizing fuzzy sets are usually symmetric triangles or trapezoids centered about representive values. (We can think of such sets as *fuzzy numbers.*) The set ZE may be a Gaussian curve for the pendulum angle θ, a triangle for the angular velocity $\Delta\theta$, and a trapezoid for the velocity v. But all the ZE fuzzy sets will be centered about the numerical value zero, which will have maximum membership in the set of zero values.

How much should contiguous fuzzy sets overlap? This design issue depends on the problem at hand. Too much overlap blurs the distinction between the fuzzy set values. Too little overlap tends to resemble bivalent control, producing overshoot and undershoot. In Chapter 19 we determine experimentally the following default heuristic for ideal overlap: *Contiguous fuzzy sets in a library should overlap approximately* 25%.

FAM rules are triples, such as $(NM, Z;\ PM)$. They describe how to modify the control variable for observed values of the pendulum state variables. A FAM rule associates a motor-velocity fuzzy set value with a pendulum-angle fuzzy set value and an angular-velocity fuzzy set value. So we can interpret the triple $(NM, Z;\ PM)$ as the set-level

implication

IF the pendulum angle θ is negative but medium
AND the angular velocity $\Delta\theta$ is about zero ,
THEN the motor velocity should be positive but medium .

These commonsensical FAM rules are comparatively easy to articulate in natural language. Consider a terser linguistic version of the same three-antecedent FAM rule:

IF $\theta = NM$ AND $\Delta\theta = ZE$,
THEN $v = PM$.

Even this mild level of formalism may inhibit the knowledge acquisition process. On the other hand, the still terser FAM triple $(NM, ZE;\ PM)$ allows knowledge to be acquired simply by filling in a few entries in a linguistic FAM-bank matrix. In practice this often allows a working system to be developed in hours, if not minutes.

We specify the pendulum FAM system when we choose a FAM bank of two-antecedent FAM rules. Perhaps the first FAM rule to choose is the *steady-state FAM rule*: $(ZE, ZE;\ ZE)$. The steady-state FAM rule describes what to do in equilibrium. For the inverted pendulum we should do nothing.

This is typical of many control problems that require nulling a scalar error measure. We can control multivariable problems by nulling the norms of the system error vector and error-velocity vectors, or, better, by directly nulling the individual scalar variables. (Chapter 19 shows how error nulling can control a realtime target tracking system.) Error nulling tractably extends the FAM methodology to nonlinear estimation, control, and decision problems of high dimension.

The pendulum FAM bank is a 7-by-7 matrix with linguistic fuzzy-set entries. We index the columns by the seven fuzzy sets that quantize the angle θ universe of discourse. We index the rows by the seven fuzzy sets that quantize the angular velocity $\Delta\theta$ universe of discourse.

Each matrix entry is one of seven motor-velocity fuzzy-set values. Since a FAM rule is a mapping or function, there is exactly one output velocity value for every pair of angle and angular-velocity values. So the 49 entries in the FAM bank matrix represent the 49 possible two-antecedent FAM rules. In practice most of the entries are blank. In the adaptive FAM case discussed below, we adaptively generate the entries from process sample data.

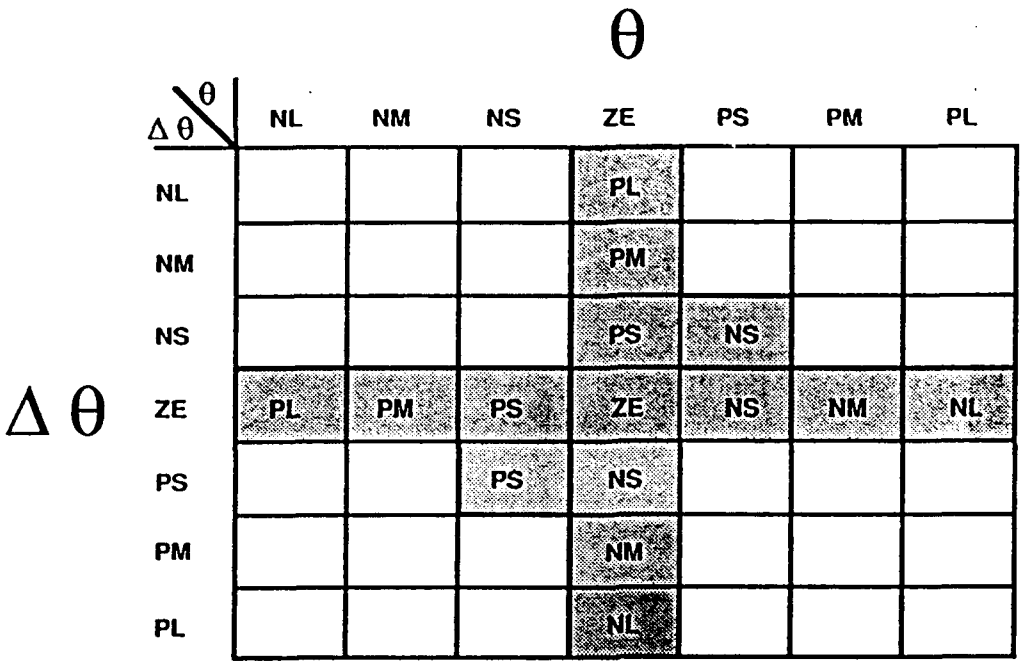
Commonsense dictates the entries in the pendulum FAM bank matrix. Suppose the pendulum is not changing. So $\Delta\theta = ZE$. If the pendulum is to the right of vertical, the motor velocity should be negative to compensate. The farther the pendulum is to the right, the larger the negative motor velocity should be. The motor velocity should be positive if the pendulum is to the left. So the fourth row of the FAM bank matrix, which corresponds to $\Delta\theta = ZE$, should be the ordinal inverse of the θ row values. This assignment includes the steady-state FAM rule $(ZE, ZE;\ ZE)$.

Now suppose the angle θ is zero but the pendulum is moving. If the angular velocity is negative, the pendulum will overshoot to the left. So the motor velocity should be positive to compensate. If the angular velocity is positive, the motor velocity should be negative. The greater the angular velocity is in magnitude, the greater the motor velocity should be in magnitude. So the fourth column of the FAM bank matrix, which corresponds to $\theta = ZE$, should be the ordinal inverse of the $\Delta\theta$ column values. This assignment also includes the steady-state FAM rule.

Positive θ values with negative $\Delta\theta$ values should produce negative motor velocity values, since the pendulum is heading toward the vertical. So $(PS, NS;\ NS)$ is a candidate FAM rule. Symmetrically, negative θ values with positive $\Delta\theta$ values should produce positive motor velocity values. So $(NS, PS;\ PS)$ is another candidate FAM rule.

This gives 15 FAM rules altogether. In practice these rules are more than sufficient to successfully balance an inverted pendulum. Different, and smaller, subsets of FAM rules may also successfully balance the pendulum.

We can represent the bank of 15 FAM rules as the 7-by-7 linguistic matrix



The BIOFAM system F also admits a geometric interpretation. The set of all possible input-outpairs $(\theta, \Delta\theta;\ F(\theta, \Delta\theta))$ defines a *FAM surface* in the input-output product space, in this case in R^3. We plot examples of these control surfaces in Chapters 18 and 19.

The BIOFAM *inference procedure* activates in parallel the antecedents of all 15 FAM rules. The binary or pulse nature of inputs picks off single fit values from the quantizing fuzzy sets. We can use either the correlation-minimum or correlation-product inferencing technique. For simplicity we shall illustrate the procedure with correlation-minimum inferencing.

Suppose the current pendulum angle θ is 15 degrees and the angular velocity $\Delta\theta$ is -10. This amounts to passing two bit vectors of one 1 and all else 0 through the BIOFAM system. What is the corresponding motor velocity value $v = F(15, -10)$?

Consider first how the input data pair (15, -10) activates steady-state FAM rule $(ZE, ZE;\ ZE)$. Suppose we define the antecedent and consequent fuzzy sets for ZE with the triangular fuzzy set membership functions in Figure 17.4. Then the angle datum 15 is a zero angle value to degree .2 : $m^{\theta}_{ZE}(15) = .2$. The angular velocity datum -10 is a zero

angular velocity value to degree .5 : $m_{ZE}^{\Delta\theta}(-10) = .5$.

We combine the antecedent fit values with minimum or maximum according as the antecedent fuzzy sets are combined with the conjunctive AND or the disjunctive OR. Intuitively, it should be at least as difficult to satisfy both antecedent conditions as to satisfy either one separately.

The FAM rule notation $(ZE, ZE;\ ZE)$ implicitly assumes that antecedent fuzzy sets are combined conjunctively with AND. So the data satisfy the compound antecedent of the FAM rule $(ZE, ZE;\ ZE)$ to degree

$$\begin{aligned} \min(m_{ZE}^{\theta}(15),\ m_{ZE}^{\Delta\theta}(-10)) &= \min(.2,\ .5) \\ &= .2 \quad . \end{aligned}$$

Clearly this methodology extends to any number of antecedent terms connected with arbitrary logical (set-theoretical) connectives.

The system should now activate the consequent fuzzy set of zero motor velocity values to degree .2. This is not the same as activating the ZE motor velocity fuzzy set 100% with probability .2, and certainly not the same as $\text{Prob}\{v = 0\} = .2$. Instead a deterministic 20% of ZE should result and, according to the additive combination formula (17), should be added to the final output fuzzy set.

The correlation-minimum inference procedure activates the angular velocity fuzzy set ZE to degree .2 by taking the pairwise minimum of .2 and the ZE fuzzy set m_{ZE}^{v}:

$$\min(m_{ZE}^{\theta}(15),\ m_{ZE}^{\Delta\theta}(-10)) \wedge m_{ZE}^{v}(v) = .2 \wedge m_{ZE}^{v}(v)$$

for all velocity values v. The correlation-product inference procedure would simply multiply the zero angular velocity fuzzy set by .2 : $.2\ m_{ZE}^{v}(v)$ for all v.

The data similarly activate the FAM rule $(PS, ZE;\ NS)$ depicted in Figure 17.4. The angle datum 15 is a small but positive angle value to degree .8. The angular velocity datum -10 is a zero angular velocity value to degree .5. So the output motor velocity fuzzy set of small but negative motor velocity values is scaled by .5, the lesser of the two antecedent fit values:

$$\min(m_{PS}^{\theta}(15),\ m_{ZE}^{\Delta\theta}(-10)) \ \wedge \ m_{NS}^{v}(v) \ = \ .5 \ \wedge \ m_{NS}^{v}(v)$$

for all velocity values v. So the data activate the FAM rule $(PS, ZE;\ NS)$ to greater degree than the steady-state FAM rule $(ZE, ZE;\ ZE)$ since in this example an angle value of 15 degrees is more a small but positive angle value than a zero angle value.

The data similarly activate the other 13 FAM rules. We combine the resulting minimum-scaled consequent fuzzy sets according to (17) by summing pointwise. We can then compute the fuzzy centroid with equation (19), with perhaps integrals replacing the discrete sums, to determine the specific output motor velocity v. In Chapter 19 we show that, for symmetric fuzzy sets of quantization, the centroid can always be computed exactly with simple discrete sums even if the fuzzy sets are continuous. In many realtime applications we must repeat this entire FAM inference procedure hundreds, perhaps thousands, of times per second. This requires fuzzy VLSI or optical processors.

Figure 17.4 illustrates this equal-weight additive combination procedure for just the FAM rules $(ZE, ZE;\ ZE)$ and $(PS, ZE;\ NS)$. The fuzzy-centroidal motor velocity value in this case is -3.

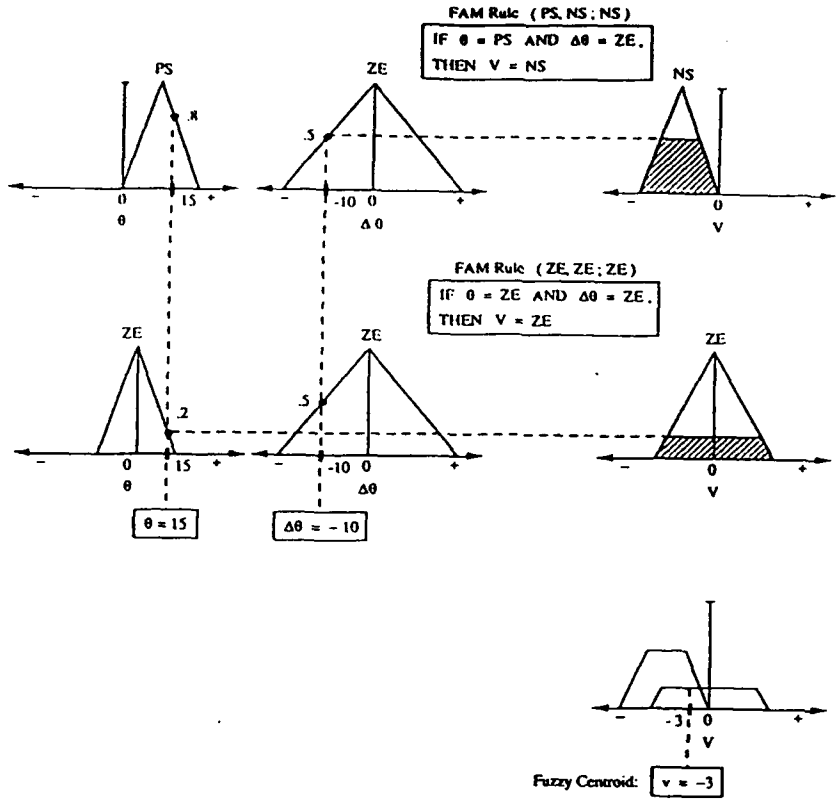


FIGURE 17.4 FAM correlation-minimum inference procedure. The FAM system consists of the two two-antecedent FAM rules $(PS, ZE;\ NS)$ and $(ZE, ZE;\ ZE)$. The input angle datum is 15, and is more a small but positive angle value than a zero angle value. The input angular velocity datum is -10, and is only a zero angular velocity value to degree .5. Antecedent fit values are combined with minimum since the antecedent terms are combined conjunctively with AND. The combined fit value then scales the consequent fuzzy set with pairwise minimum. The minimum-scaled output fuzzy sets are added pointwise. The fuzzy centroid of this output waveform is computed and yields the system output velocity value -3.

Multi-Antecedent FAM Rules: Decompositional Inference

The BIOFAM inference procedure treats antecedent fuzzy sets as if they were propositions with fuzzy truth values. This is because fuzzy logic corresponds to 1-dimensional

fuzzy set theory and because we use binary or exact inputs. We now formally develop the connection between BIOFAMs and the FAM theory presented earlier.

Consider the compound FAM rule "IF X is A AND Y is B , THEN C is Z," or $(A, B;\ C)$ for short. Let the universes of discourse X, Y, and Z have dimensions n, p, and q: $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_p\}$, and $Z = \{z_1, \ldots, z_q\}$. We can directly extend this framework to multiple antecedent and consequent terms.

In our notation X, Y, and Z are both universes of discourse and fuzzy variables. The fuzzy *variable* X can assume the fuzzy set *values* $A_1, A_2, \ldots$, and similarly for the fuzzy variables Y and Z. When controlling an inverted pendulum, the identification "X is A" might represent the natural-language description "The pendulum angle is positive but small."

What is the matrix representation of the FAM rule $(A, B;\ C)$? The question is nontrivial since A, B, and C are fuzzy subsets of different universes of discourse, points in different unit cubes. Their dimensions and interpretations differ. Mamdani [1977] and others have suggested representing such rules as fuzzy multidimensional relations or arrays. Then the FAM rule $(A, B;\ C)$ would be a fuzzy subset of the product space $X \times Y \times Z$. This representation is not used in practice since only exact inputs are presented to FAM systems and the BIOFAM procedure applies. If we presented the system with a genuine fuzzy set input, we would no doubt preprocess the fuzzy set with a centroidal or maximum-fit-value technique so we could still apply the BIOFAM inference procedure.

We present an alternative representation that decomposes, then recomposes, the FAM rule $(A, B;\ C)$ in accord with the FAM inference procedure. This representation allows neural networks to adaptively estimate, store, and modify the decomposed FAM rules. The representation requires far less storage than the multidimensional-array representation.

Let the fuzzy Hebb matrices M_{AC} and M_{BC} store the simple FAM associations (A, C) and (B, C):

$$M_{AC} = A^T \circ C \quad , \tag{20}$$

$$M_{BC} = B^T \circ C \quad . \tag{21}$$

The fuzzy Hebb matrices M_{AC} and M_{BC} *split* the compound FAM rule $(A, B;\ C)$. We can construct the splitting matrices with correlation-product encoding.

Let $I_X^i = (0 \ldots 0\ 1\ 0 \ldots 0)$ be an n-dimensional bit vector with ith element 1 and all other elements 0. I_X^i is the ith row of the n-by-n identity matrix. Similarly, I_Y^j and I_Z^k are the respective jth and kth rows of the p-by-p and q-by-q identity matrices. The bit vector I_X^i represents the occurrence of the exact input x_i.

We will call the proposed FAM representation scheme **FAM decompositional inference**, in the spirit of the max-min compositional inference scheme discussed above. FAM decompositional inference *decomposes* the compound FAM rule $(A, B;\ C)$ into the component rules (A, C) and (B, C). The simpler component rules are processed in parallel. New fuzzy set inputs A' and B' pass through the FAM matrices M_{AC} and M_{BC}. Max-min composition then gives the recalled fuzzy sets $C_{A'}$ and $C_{B'}$:

$$C_{A'} = A' \circ M_{AC} \quad , \tag{22}$$

$$C_{B'} = B' \circ M_{BC} \quad . \tag{23}$$

The trick is to *recompose* the fuzzy sets $C_{A'}$ and $C_{B'}$ with intersection or union according as the antecedent terms "X is A" and "Y is B" are combined with AND or OR. The negated antecedent term "X is NOT A" requires forming the set complement $C_{A'}^c$ for input fuzzy set A'.

Suppose we present the new inputs A' and B' to the single-FAM-rule system F that stores the FAM rule $(A, B;\ C)$. Then the recalled output fuzzy set C' equals the intersection of $C_{A'}$ and $C_{B'}$:

$$\begin{aligned} F(A',\ B') &= [A' \circ M_{AC}] \cap [B' \circ M_{BC}] \\ &= C_{A'} \cap C_{B'} \\ &= C' \quad . \end{aligned} \tag{24}$$

We can then defuzzify C', if we wish, to yield the exact output I_Z^k.

The logical connectives apply to the antecedent terms of different dimension and meaning. Decompositional inference applies the set-theoretic analogues of the logical connectives to subsets of Z. Of course all subsets C' of Z have the same dimension and meaning.

We now prove that decompositional inference generalizes BIOFAM inference. This generalization is not simply formal. It opens an immediate path to adaptation with arbitrary neural network techniques.

Suppose we present the exact inputs x_i and y_j to the single-FAM-rule system F that stores $(A, B;\ C)$. So we present the unit bit vectors I_X^i and I_Y^j to F as nonfuzzy set inputs. Then

$$\begin{aligned} F(x_i, y_j) \;=\; F(I_X^i,\ I_Y^j) \;&=\; [I_X^i \circ M_{AC}] \;\cap\; [I_Y^j \circ M_{BC}] \\ &=\; a_i \wedge C \;\cap\; b_j \wedge C \qquad (25) \\ &=\; \min(a_i,\ b_j) \wedge C \quad . \qquad (26) \end{aligned}$$

(25) follows from (8). Representing C with its membership function m_C, (26) is equivalent to the BIOFAM prescription

$$\min(a_i,\ b_j) \wedge m_C(z) \qquad (27)$$

for all z in Z.

If we encode the simple FAM rules (A, C) and (B, C) with correlation-product encoding, decompositional inference gives the BIOFAM version of correlation-product inference:

$$\begin{aligned} F(I_X^i, I_Y^j) \;&=\; [I_X^i \circ A^T C] \;\cap\; [I_Y^j \circ B^T C] \\ &=\; a_i\, C \;\cap\; b_j\, C \qquad (28) \\ &=\; \min(a_i,\ b_j)\, C \qquad (29) \\ &=\; \min(a_i,\ b_j)\, m_C(z) \qquad (30) \end{aligned}$$

for all z in Z. (13) implies (28). $\min(a_i\, c_k,\ b_j\, c_k) \;=\; \min(a_i,\ b_j)\, c_k$ implies (29).

Decompositional inference allows arbitrary fuzzy sets, waveforms, or distributions A' and B' to be applied to a FAM system. The FAM system can house an arbitrary FAM bank of compound FAM rules. If we use the FAM system to control a process, the input fuzzy sets A' and B' can be the output of an independent state-*estimation* system, such as a Kalman filter. A' and B' might then represent probability distributions on the exact input spaces X and Y. The filter-controller cascade is a common engineering architecture.

We can split compound consequents as desired. We can split the compound FAM rule "IF X is A AND Y is B, THEN Z is C OR W is D," or $(A, B;\ C, D)$, into the FAM rules $(A, B;\ C)$ and $(A, B;\ D)$. We can use the same split if the consequent logical connective is AND.

We can give a propositional-calculus justification for the decompositional inference technique. Let A, B, and C be bivalent propositions with truth values $t(A)$, $t(B)$, and $t(C)$ in $\{0, 1\}$. Then we can construct truth tables to prove the two consequent-splitting tautologies that we use in decompositional inference:

$$[A \longrightarrow (B \text{ OR } C)] \longrightarrow [(A \longrightarrow B) \text{ OR } (A \longrightarrow C)] , \tag{31}$$

$$[A \longrightarrow (B \text{ AND } C)] \longrightarrow [(A \longrightarrow B) \text{ AND } (A \longrightarrow C)] , \tag{32}$$

where the arrow represents logical implication.

In bivalent logic, the implication $A \rightarrow B$ is false iff the antecedent A is true and the consequent B is false. Equivalently, $t(A \rightarrow B) = 1$ iff $t(A) = 1$ and $t(B) = 0$. This allows a "brief" truth table to be constructed to check for validity. We chose truth values for the terms in the consequent of the overall implication (31) or (32) to make the consequent false. Given those restrictions, if we cannot find truth values to make the antecedent true, the statement is a tautology. In (31), if $t((A \rightarrow B) \text{ OR } (A \rightarrow C)) = 0$, then $t(\cdot) = 1$ and $t(B) = t(C) = 0$, since a disjunction is false iff both disjuncts are false. This forces the antecedent $A \rightarrow (B \text{ OR } C)$ to be false. So (31) is a **tautology**: It is true in all cases.

We can also justify splitting the compound FAM rule "IF X is A OR Y is B, THEN Z is C" into the disjunction (union) of the two simple FAM rules "IF X is A,

THEN Z is C " and "IF Y is B , THEN Z is C " with a propositional tautology:

$$[(A \text{ OR } B) \longrightarrow C] \longrightarrow [(A \longrightarrow C) \text{ OR } (B \longrightarrow C)] \ . \tag{33}$$

Now consider splitting the original compound FAM rule "IF X is A AND Y is B , THEN Z is C " into the conjunction (intersection) of the two simple FAM rules "IF X is A , THEN Z is C " and "IF Y is B , THEN Z is C ." A problem arises when we examine the truth table of the corresponding proposition

$$[(A \text{ AND } B) \longrightarrow C] \longrightarrow [(A \longrightarrow C) \text{ AND } (B \longrightarrow C)] \ . \tag{34}$$

The problem is that (34) is not always true, and hence not a tautology. The implication is false if A is true and B and C are false, or if A and C are false and B is true. But the implication (34) is valid if *both* antecedent terms A and B are true. So if $t(A) = t(B) = 1$, the compound conditional $(A \text{ AND } B) \rightarrow C$ implies both $A \rightarrow C$ and $B \rightarrow C$.

The simultaneous occurrence of the data values x_i and y_j satisfies this condition. Recall that logic is 1-dimensional set theory. The condition $t(A) = t(B) = 1$ is given by the 1 in I_X^i and the 1 in I_X^j. We can interpret the unit bit vectors I_X^i and I_Y^j as the (true) bivalent propositions "X is x_i" and "Y is y_j." Propositional logic applies coordinate-wise. A similar argument holds for the converse of (33).

For general fuzzy set inputs A' and B' the argument still holds in the sense of continuous-valued logic. But the truth values of the logical implications may be less than unity while greater than zero. If A' is a null vector and B' is not, or vice versa, the implication (34) is false coordinate-wise, at least if one coordinate of the non-null vector is unity. But in this case the decompositional inference scheme yields an output null vector C'. In effect the FAM system indicates the propositional falsehood.

Adaptive Decompositional Inference

The decompositional inference scheme allows the *splitting matrices* M_{AC} and M_{BC} to

be arbitrary. Indeed it allows them to be eliminated altogether.

Let $N_X: \ I^n \rightarrow I^q$ be an arbitrary *neural network* system that maps fuzzy subsets A' of X to fuzzy subsets C' of Z. $N_Y: \ I^p \rightarrow I^q$ can be a different neural network. In general N_X and N_Y are time-varying.

The adaptive decompositional inference (ADI) scheme allows compound FAM rules to be adaptively split, stored, and modified by arbitrary neural networks. The compound FAM rule "IF X is A AND Y is B, THEN Z is C," or $(A, B;\ C)$, can be split by N_X and N_Y. N_X can house the simple FAM association (A, C). N_Y can house (B, C). Then for arbitrary fuzzy set inputs A' and B', ADI proceeds as before for an adaptive FAM system $F: \ I^n \times I^p \rightarrow I^q$ that houses the FAM rule $(A, B;\ C)$ or a bank of such FAM rules:

$$\begin{aligned} F(A', B') &= N_X(A') \cap N_Y(B') \\ &= C_{A'} \cap C_{B'} \\ &= C' \quad . \end{aligned} \tag{35}$$

Any neural network technique can be used. A reasonable candidate for many unstructured problems is the backpropagation algorithm applied to several small feedforward multilayer networks. The primary concerns are space and training time. Several small neural networks can often be trained in parallel faster, and more accurately, than a single large neural network.

The ADI approach illustrates one way neural algorithms can be embedded in a FAM architecture. Below we discuss another way that uses unsupervised clustering algorithms.

ADAPTIVE FAMs: PRODUCT-SPACE CLUSTERING IN FAM CELLS

An **adaptive FAM (AFAM)** is a time-varying mapping between fuzzy cubes. In principle the adaptive decompositional inference technique generates AFAMs. But we

shall reserve the label AFAM for systems that generate FAM rules from training data but that do not require splitting and recombining FAM data.

We propose a geometric AFAM procedure. The procedure adaptively clusters training samples in the FAM system *input-output product space*. FAM mappings are balls or clusters in the input-output product space. These clusters are simply the fuzzy Hebb matrices discussed above. The procedure "blindly" generates weighted FAM rules from training data. Further training modifies the weighted set of FAM rules. We call this unsupervised procedure **product-space clustering**.

Consider first a discrete 1-dimensional FAM system $S: \ I^n \rightarrow I^p$. Then a FAM rule has the form "IF X is A_i, THEN Y is B_i" or (A_i, B_i). The input-output product space is $I^n \times I^p$.

What does the FAM rule (A_i, B_i) look like in the product space $I^n \times I^p$? It looks like a cluster of points centered at the numerical point (A_i, B_i). The FAM system maps points A near A_i to points B near B_i. The closer A is to A_i, the closer the point (A, B) is to the point (A_i, B_i) in the product space $I^n \times I^p$. In this sense FAMs map balls in I^n to balls in I^p. The notation is ambiguous since (A_i, B_i) stands for both the FAM rule mapping, or fuzzy subset of $I^n \times I^p$, and the numerical fit-vector point in $I^n \times I^p$.

Adaptive clustering algorithms can estimate the unknown FAM rule (A_i, B_i) from training samples of the form (A, B). In general there are m unknown FAM rules $(A_1, B_1), \ldots, (A_m, B_m)$. The number m of FAM rules is also unknown. The user may select m arbitrarily in many applications.

Competitive *adaptive vector quantization* (AVQ) algorithms can adaptively estimate both the unknown FAM rules (A_i, B_i) and the unknown number m of FAM rules from FAM system input-output data. The AVQ algorithms do not require fuzzy-set data. Scalar BIOFAM data suffices, as we illustrate below for adaptive estimation of inverted-pendulum control FAM rules.

Suppose the r fuzzy sets $A_1, \ldots, A_r$ quantize the input universe of discourse X. The s fuzzy sets $B_1, \ldots, B_s$ quantize the output universe of discourse Y. In general r and s are unrelated to each other and to the number m of FAM rules (A_i, B_i). The user must specify r and s and the shape of the fuzzy sets A_i and B_i. In practice this is not difficult.

Quantizing fuzzy sets are usually trapezoidal, and r and s are less than 10.

The quantizing collections $\{A_i\}$ and $\{B_j\}$ define rs FAM cells F_{ij} in the input-output product space $I^n \times I^p$. The FAM cells F_{ij} overlap since contiguous quantizing fuzzy sets A_i and A_{i+1}, and B_j and B_{j+1}, overlap. So the FAM cell collection $\{F_{ij}\}$ does not partition the product space $I^n \times I^p$. The union of all FAM cells also does not equal $I^n \times I^p$ since the patches F_{ij} are fuzzy subsets of $I^n \times I^p$. The union provides only a fuzzy "cover" for $I^n \times I^p$.

The *fuzzy Cartesian product* $A_i \times B_i$ defines the FAM cell F_{ij}. $A_i \times B_i$ is just the fuzzy outer product $A_i^T \circ B_i$ in (6) or the correlation product $A_i^T \; B_i$ in (12). So a FAM cell F_{ij} is simply the fuzzy correlation-minimum or correlation-product matrix M_{ij} : $F_{ij} = M_{ij}$.

Adaptive FAM Rule Generation

Let $\mathbf{m}_1, \ldots, \mathbf{m}_k$ be k quantization vectors in the input-output product space $I^n \times I^p$ or, equivalently, in I^{n+p}. $\mathbf{m}_j$ is the jth column of the synaptic connection matrix $\mathbf{M}$. $\mathbf{M}$ has $n + p$ rows and k columns.

Suppose, for instance, $\mathbf{m}_j$ changes in time according to the differential competitive learning (DCL) AVQ algorithm discussed in Chapters 6 and 9. The competitive system samples concatenated fuzzy set samples of the form $[A|B]$. The augmented fuzzy set $[A|B]$ is a point in the unit hypercube I^{n+p}.

The synaptic vectors $\mathbf{m}_j$ converge to FAM matrix centroids in $I^n \times I^p$. More generally they estimate the density or distribution of the FAM rules in $I^n \times I^p$. The quantizing synaptic vectors naturally weight the estimated FAM rule. The more synaptic vectors clustered about a centroidal FAM rule, the greater its weight w_i in (17).

Suppose there are 15 FAM-rule centroids in $I^n \times I^p$ and $k > 15$. Suppose k_i synaptic vectors $\mathbf{m}_j$ cluster around the ith centroid. So $k_1 + \ldots + k_{15} = k$. Suppose the *cluster counts* k_i are ordered as

$$k_1 \geq k_2 \geq \ldots k_{15} \quad . \tag{36}$$

The first centroidal FAM rule is as at least as frequent as the second centroidal FAM rule, and so on. This gives the adaptive FAM-rule weighting scheme

$$w_i = \frac{k_i}{k} \quad . \tag{37}$$

The FAM rule weights w_i evolve in time as new augmented fuzzy sets $[A|B]$ are sampled. In practice we may want only the 15 most-frequent FAM rules or only the FAM rules with at least some minimum frequency $w_{\min}$. Then (37) provides a quantitative solution.

Geometrically we count the number k_{ij} of quantizing vectors in each FAM cell F_{ij}. We can define FAM-cell boundaries in advance. High-count FAM cells outrank low-count FAM cells. Most FAM cells contain zero or few synaptic vectors.

Product-space clustering extends to compound FAM rules and product spaces. The FAM rule "IF X is A AND Y is B, THEN Z is C", or $(A,\ B;\ C)$, is a point in $I^n \times I^p \times I^q$. The t fuzzy sets $C_1, \ldots, C_t$ quantize the new output space Z. There are rst FAM cells F_{ijk}. (36) and (37) extend similarly. X, Y, and Z can be continuous. The adaptive clustering procedure extends to any number of FAM-rule antecedent terms.

Adaptive BIOFAM Clustering

BIOFAM data clusters more efficiently than fuzzy-set FAM data. Paired numbers are easier to process and obtain than paired fit vectors. This allows system input-output data to directly generate FAM systems.

In control applications, human or automatic controllers generate streams of "well-controlled" system input-output data. Adaptive BIOFAM clustering converts this data to weighted FAM rules. The adaptive system transduces behavioral data to behavioral rules. The fuzzy system learns causal patterns. It learns which control inputs cause which control outputs. The system approximates these causal patterns when it acts as the controller.

Adaptive BIOFAMs cluster in the input-output product space $X \times Y$. The product space $X \times Y$ is vastly smaller than the power-set product space $I^n \times I^p$ used above. The

adaptive synaptic vectors $\mathbf{m}_j$ are now 2-dimensional instead of $n + p$-dimensional. On the other hand, competitive BIOFAM clustering requires many more input-output data pairs $(x_i, y_i) \in R^2$ than augmented fuzzy-set samples $[A|B] \in I^{n+p}$.

Again our notation is ambiguous. We now use x_i as the numerical sample from X at sample time i. Earlier x_i denoted the ith ordered element in the finite nonfuzzy set $X = \{x_1, \ldots, x_n\}$. One advantage is X can be continuous, say R^n.

BIOFAM clustering counts synaptic quantization vectors in FAM cells. The system samples the nonfuzzy input-output stream $(x_1, y_1), (x_2, y_2), \ldots$ Unsupervised competitive learning distributes the k synaptic quantization vectors $\mathbf{m}_1, \ldots, \mathbf{m}_k$ in $X \times Y$. Learning distributes them to different FAM cells F_{ij}. The FAM cells F_{ij} overlap but are nonfuzzy subcubes of $X \times Y$. The BIOFAM FAM cells F_{ij} cover $X \times Y$.

F_{ij} contains k_{ij} quantization vectors at each sample time. The cell counts k_{ij} define a frequency *histogram* since all k_{ij} sum to k. So $w_{ij} = \frac{k_{ij}}{k}$ weights the FAM rule "IF X is A_i, THEN Y is B_j."

Suppose the pairwise-overlapping fuzzy sets NL, NM, NS, ZE, PS, PM, PL quantize the input space X. Suppose seven similar fuzzy sets quantize the output space Y. We can define the fuzzy sets arbitrarily. In practice they are normal and trapezoidal. (The boundary fuzzy sets NL and PL are ramp functions.) X and Y may each be the real line. A typical FAM rule is "IF X is NL, THEN Y is PS."

Input datum x_i is nonfuzzy. When $X = x_i$ holds, the relations $X = NL, \ldots, X = PL$ hold to different degrees. Most hold to degree zero. $X = NM$ holds to degree $m_{NM}(x_i)$. Input datum x_i partially activates the FAM rule "IF X is NM, THEN Y is ZE" or, equivalently, $(NM;\ ZE)$. Since the FAM rules have single antecedents, x_i activates the consequent fuzzy set ZE to degree $m_{NM}(x_i)$ as well. Multi-antecedent FAM rules activate output consequent sets according to a logic-based function of antecedent term membership values, as discussed above on BIOFAM inference.

Suppose Figure 17.5 represents the input-output data stream $(x_1, y_1), (x_2, y_2), \ldots$ in the planar product space $X \times Y$:

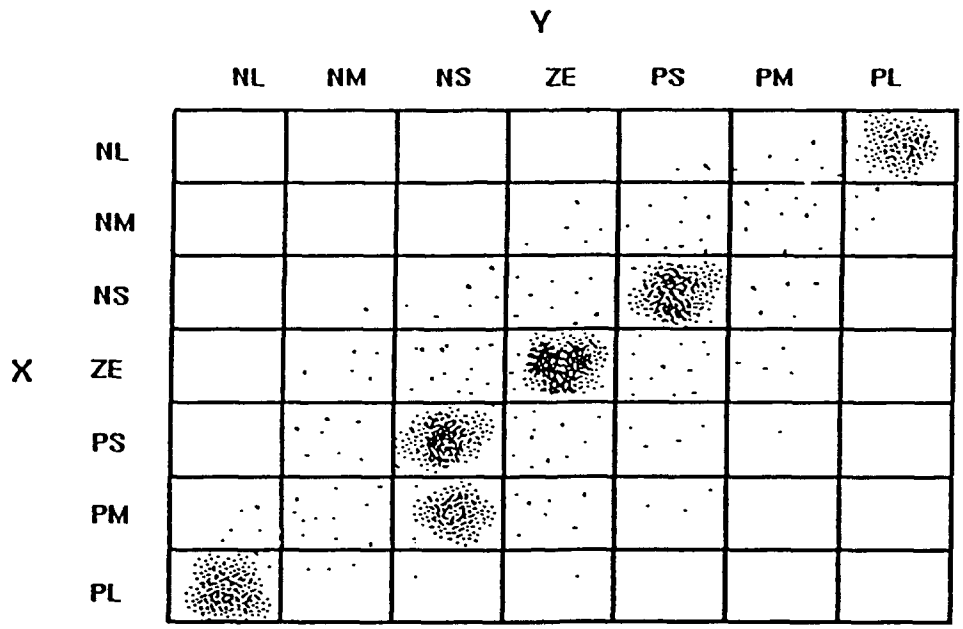


FIGURE 17.5 Distribution of input-output data (x_i, y_i) in the input-output product space $X \times Y$. Data clusters reflect FAM rules, such as the steady-state FAM rule "IF X is ZE, THEN Y is ZE".

Suppose the sample data in Figure 17.5 trains a DCL system. Suppose such competitive learning distributes ten 2-dimensional synaptic vectors $\mathbf{m}_1, \ldots, \mathbf{m}_{10}$ as in Figure 17.6:

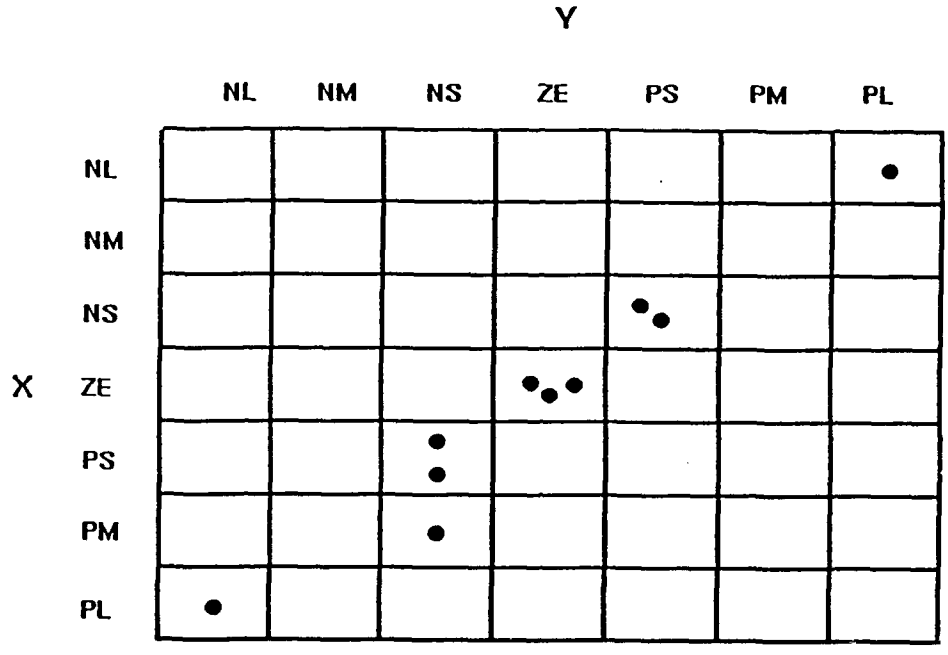


FIGURE 17.6 Distribution of ten 2-dimensional synaptic quantization vectors $\mathbf{m}_1, \ldots, \mathbf{m}_{10}$ in the input-output product space $X \times Y$. As the FAM system samples nonfuzzy data (x_i, y_i), competitive learning distributes the synaptic vectors in $X \times Y$. The synaptic vectors estimate the frequency distribution of the sampled input-output data, and thus estimate FAM rules.

FAM cells do not overlap in Figures 17.5 and 17.6 for convenience's sake. The corresponding quantizing fuzzy sets touch but do not overlap.

Figure 17.5 reveals six sample-data clusters. The six quantization-vector clusters in Figure 17.6 estimate the six sample-data clusters. The single synaptic vector in FAM cell $(PM;\ NS)$ indicates a smaller cluster. Since $k = 10$, the number of quantization vectors in each FAM cell measures the percentage or frequency weight w_{ij} of each possible FAM rule.

In general the additive combination rule (17) does not require normalizing the quantization-vector count k_{ij}. $w_{ij} = k_{ij}$ is acceptable. This holds for both maximum-membership defuzzification (18) and fuzzy centroid defuzzification (19). These defuzzification schemes prohibit only negative weight values.

The ten quantization vectors in Figure 17.6 estimate at most six FAM rules. From most to least frequent or "important", the FAM rules are $(ZE;\ ZE)$, $(PS;\ NS)$, $(NS;\ PS)$, $(PM;\ NS)$, $(PL;\ NL)$, and $(NL;\ PL)$. These FAM rules suggest that fuzzy variable X is an error variable or an error velocity variable since the steady-state FAM rule $(ZE;\ ZE)$ is most important. If we sample a system only in steady-state equilibrium, we will estimate only the steady-state FAM rule. We can accurately estimate the FAM system's global behavior only if we representatively sample the system's input-output behavior.

The "corner" FAM rules $(PL;\ NL)$ and $(NL;\ PL)$ may be more important than their frequencies suggest. The boundary sets Negative Large (NL) and Positive Large (PL) are usually defined as ramp functions, as negatively and positively sloped lines. NL and PL alone cover the important end-point regions of the universe of discourse X. They give $m_{NL}(x) = m_{PL}(x) = 1$ only if x is at or near the end-point of X, since NL and PL are ramp functions not trapezoids. NL and PL cover these end-point regions "briefly". Their corresponding FAM cells tend to be smaller than the other FAM cells. The end-point regions must be covered in most control problems, especially error nulling problems like stabilizing an inverted pendulum. The user can weight these FAM-cell counts more highly, for instance $w_{ij} = c\, k_{ij}$ for scaling constant $c > 0$. Or the user can simply include these end-point FAM rules in every operative FAM bank.

Most FAM cells do not generate FAM rules. More accurately, we estimate every possible FAM rule but usually with zero or near-zero frequency weight w_{ij}. For large numbers of multiple FAM-rule antecedents, system input-output data streams through comparatively few FAM cells. Structured trajectories in $X \times Y$ are few.

A FAM-rule's mapping structure also limits the number of estimated FAM rules. A FAM rule maps fuzzy sets in I^n or $F(2^X)$ to fuzzy sets in I^p or $F(2^Y)$. A fuzzy associative memory maps every domain fuzzy set A to a unique range fuzzy set B. Fuzzy set A cannot map to multiple fuzzy sets B, B', B'', and so on. We write the FAM rule as $(A;\ B)$ not

(A B or B' or B'' or...). So we estimate *at most* one rule per FAM-cell row in Figure 17.6.

If two FAM cells in a row are equally and highly frequent, we can pick arbitrarily either FAM rule to include in the FAM bank. This occurs infrequently but can occur. In principle we could estimate the FAM rule as a compound FAM rule with a disjunctive consequent. The simplest strategy picks only the highest frequency FAM cell per row.

The user can estimate FAM rules without counting the quantization vectors in each FAM cell. There may be too many FAM cells to search at each estimation iteration. The user never need examine FAM cells. Instead the user checks the synaptic vector components m_{ij}. The user defines in advance fuzzy-set intervals, such as $[l_{NL}, u_{NL}]$ for NL. If $l_{NL} \leq m_{ij} \leq u_{NL}$, then the FAM-antecedent reads "IF X is NL."

Suppose the input and output spaces X and Y are the same, the real interval $[-35, 35]$. Suppose we partition X and Y into the same seven disjoint fuzzy sets:

NL = [-35, -25]
NM = [-25, -15]
NS = [-15, -5]
ZE = [-5, 5]
PS = [5, 15]
PM = [15, 25]
PL = [25, 35] .

Then the observed synaptic vector $\mathbf{m}_j = [9, -10]$ increases the count of FAM cell $PS \times NS$ and increases the weight of FAM rule "IF X is PS, THEN Y is NS."

This amounts to nearest-neighbor classification of synaptic quantization vectors. We assign quantization vector $\mathbf{m}_k$ to FAM cell F_{ij} iff $\mathbf{m}_k$ is closer to the centroid of F_{ij} than to all other FAM-cell centroids. We break ties arbitrarily. Centroid classification allows the FAM cells to overlap.

Adaptive BIOFAM Example: Inverted Pendulum

We used DCL to train an AFAM to control the inverted pendulum discussed above. We used the accompanying C-software to generate 1,000 pendulum trajectory data. These product-space training vectors $(\theta,\ \Delta\theta,\ v)$ were points in R^3. Pendulum angle θ data ranged between -90 and 90. Pendulum angular veclocity $\Delta\theta$ data ranged from -150 to 150.

We defined FAM cells by uniformly partitioning the effective product space. Fuzzy variables could assume only the five fuzzy set values NM, NS, ZE, PS, and PM. So there were 125 possible FAM rules. For instance, the steady-state FAM rule took the form $(ZE,\ ZE;\ ZE)$ or, more completely, "IF $\theta = ZE$ AND $\Delta\theta = ZE$, THEN $v = ZE$."

A BIOFAM controlled the inverted pendulum. The BIOFAM restored the pendulum to equilibrium as we knocked it over to the right and to the left. (Function keys F9 and F10 knock the pendulum over to the left and to the right. Input-output sample data reads automatically to a training data file.) Eleven FAM rules described the BIOFAM controller. Figure 17.1 displays this FAM bank. Observe that the zero (ZE) row and column are ordinal inverses of the respective row and column indices.

Δθ \ θ	NM	NS	Z	PS	PM
NM			PM		
NS			PS	Z	
Z	PM	PS	Z	NS	NM
PS		Z	NS		
PM			NM		

FIGURE 17.7 Inverted-pendulum FAM bank used in simulation. This

BIOFAM generated 1,000 sample vectors of the form $(\theta,\ \Delta\theta,\ v)$.

We trained 125 3-dimensional synaptic quantization vectors with differential competitive learning, as discussed in Chapters 4,6, and 9. In principle the 125 synaptic vectors could describe a uniform distribution of product-space trajectory data. Then the 125 FAM cells would each contain one synaptic vector. Alternatively, if we used a vertically stabilized pendulum to generate the 1,000 training vectors, all 125 synaptic vectors would concentrate in the $(ZE,\ ZE;\ ZE)$ FAM cell. This would still be true if we only mildly perturbed the pendulum from vertical equilibrium.

DCL distributed the 125 synaptic vectors to 13 FAM cells. So we estimated 13 FAM rules. Some FAM cells contained more synaptic vectors than others. Figure 17.8 displays the synaptic-vector histogram after the DCL samples the 1,000 samples. Actually Figure 17.8 displays a truncated histogram. The horizontal axis should list all 125 FAM cells, all 125 FAM-rule weights w_k in (17). The missing 112 entries have zero synaptic-vector frequency.

Figure 17.8 gives a snapshot of the adaptive process. In practice, and in principle, successive data gradually modify the histogram. "Good" training samples should include a significant number of equilibrium samples. In Figure 17.8 the steady-state FAM cell $(ZE,\ ZE;\ ZE)$ is clearly the most frequent.

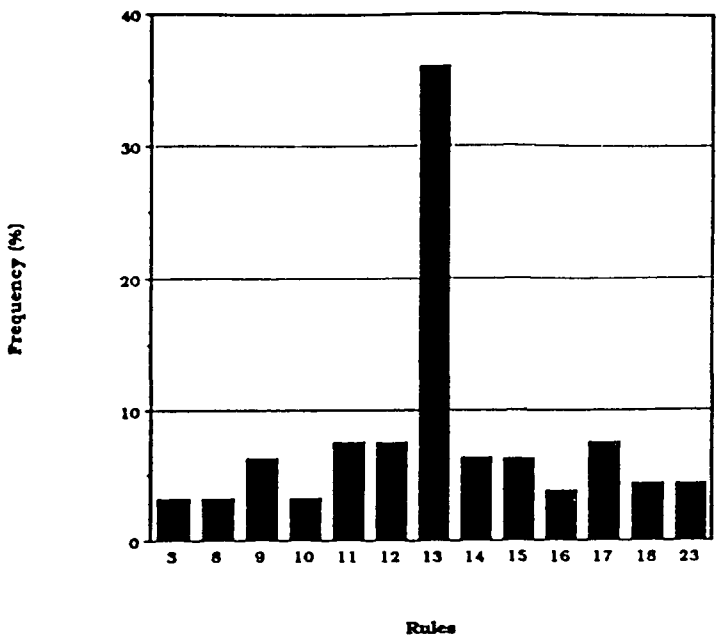


FIGURE 17.8 Synaptic-vector histogram. Differential competitive learning allocated 125 3-dimensional synaptic vectors to the 125 FAM cells. Here the adaptive system has sampled 1,000 representative pendulum-control data. DCL allocates the synaptic vectors to only 13 FAM cells. The steady-state FAM cell (ZE, ZE; ZE) is most frequent.

Figure 17.9 displays the DCL-estimated FAM bank. The product-space clustering method rapidly recovered the 11 original FAM rules. It also estimated the two additional FAM rules (PS, NM; ZE) and (NS, PM; ZE), which did not affect the BIOFAM system's performance. The estimated FAM bank defined a BIOFAM, with all 13 FAM-rule weights set w_k equal to unity, that controlled the pendulum as well as the original BIOFAM did.

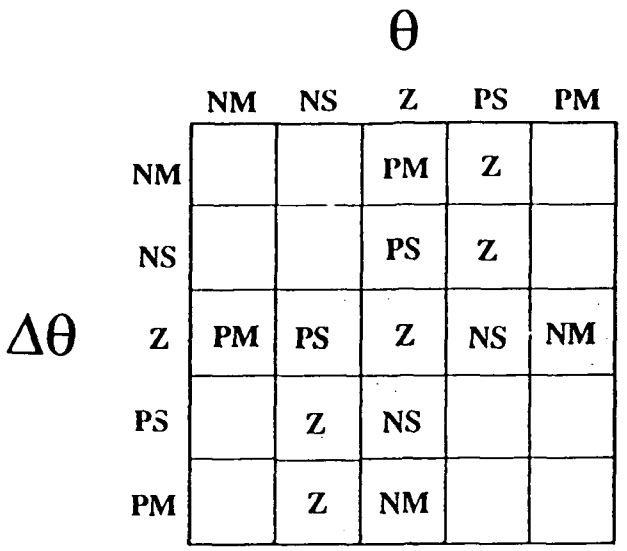


FIGURE 17.9 DCL-estimated FAM bank. Product-space clustering recovered the original 11 FAM rules and estimated two new FAM rules. The new and original BIOFAM systems controlled the inverted pendulum equally well.

In nonrealtime applications we can in principle omit the adaptive step altogether. We can directly compute the FAM-cell histogram if we exhaustively count all sampled data. Then the (growing) number of synaptic vectors equals the number of training samples. This procedure equally weights all samples, and so tends not to "track" an evolving process. Competitive learning weights more recent samples more heavily. Competitive learning's metrical-classification step also helps filter noise from the stream of sample data.

REFERENCES

Dubois, D., Prade, H., *Fuzzy Sets and Systems: Theory and Applications*, Academic Press, New York, 1980.

Hebb, D. *The Organization of Behavior*, Wiley, 1949.

Klir, G.J., Foger, T.A., *Fuzzy Sets, Uncertainty, and Information*, Prentice-Hall, 1988.

Kosko, B., "Fuzzy Knowledge Combination," *International Journal of Intelligent Systems*, vol. 1, 293 - 320, 1986.

Kosko, B., "Fuzzy Associative Memories," *Fuzzy Expert Systems*, A. Kandel (ed.), Addison-Wesley, in press, December 1986.

Kosko, B., "Fuzzy Entropy and Conditioning," *Information Sciences*, vol. 40, 165 - 174, 1986.

Kosko, B., *Foundations of Fuzzy Estimation Theory*, Ph.D. dissertation, Department of Electrical Engineering, University of California at Irvine, June 1987; Order Number 8801936, University of Microfilms International, 300 N. Zeeb Road, Ann Arbor, Michigan 48106.

Kosko, B., "Hidden Patterns in Combined and Adaptive Knowledge Networks," *International Journal of Approximate Reasoning*, vol. 2, no. 4, 377 - 393, October 1988.

Mamdani, E.H., "Application of Fuzzy Logic to Approximate Reasoning Using Linguistic Synthesis," *IEEE Transactions on Computers*, vol. C-26, no. 12, 1182 - 1191, December 1977.

Taber, W.R., Siegel, M.A., "Estimation of Expert Weights Using Fuzzy Cognitive Maps," *Proceedings of the IEEE 1st International Conference on Neural Networks (ICNN-87)*, vol. II, 319 - 325, June 1987.

Taber, W.R., "Knowledge Processing with Fuzzy Cognitive Maps," *Expert Systems with Applications*, in press, 1990.

Togai, M., Watanabe, H., "Expert System on a Chip: An Engine for Realtime Approximate Reasoning, *IEEE Expert*, vol. 1, no. 3, 1986.

Yamakawa, T., "A Simple Fuzzy Computer Hardware System Employing MIN & MAX Operations," *Proceedings of the Second International Fuzzy Systems Association (IFSA)*, Tokyo, 827 - 830, July 1987.

Yamakawa, T., "Fuzzy Microprocessors–Rule Chip and Defuzzification Chip," *Proceedings of the International Workshop on Fuzzy Systems Applications*, Iizuka-88, Kyushu Institute of Technology, 51 - 52, August 1988.

Zadeh, L.A., "A Computational Approach to Fuzzy Quantifiers in Natural Languages," *Computers and Mathematics*, vol. 9, no. 1, 149 - 184, 1983.

PROBLEMS

1. Use correlation-minimum encoding to construct the FAM matrix M from the fit-vector pair (A, B) if $A = (.6\ 1\ .2\ .9)$ and $B = (.8\ .3\ 1)$. Is (A, B) a bidirectional fixed point? Pass $A' = (.2\ .9\ .3\ .2)$ through M and $B' = (.9\ .5\ 1)$ through M^T. Do the recalled fuzzy sets differ from B and A?

2. Repeat Problem 1 using correlation-product encoding.

3. Compute the fuzzy entropy $E(M)$ of M in Problems 1 and 2.

4. If $M = A^T \circ B$ in Problem 1, find a different FAM matrix M' with greater fuzzy entropy, $E(M') > E(M)$, but that still gives perfect recall: $A \circ M' = B$. Find the *maximum entropy fuzzy associative memory* (MEFAM) matrix $\mathbf{M}^*$ such that $A \circ \mathbf{M}^* = B$.

5. Prove: If $M = A^T \circ B$ or $M = A^T B, A \circ M = B$, and $A \subset A'$, then $A' \circ M = B$.

6. Prove: $\max\limits_{1 \le k \le m} \min(a_k, b_k) \le \min(\max\limits_{1 \le k \le m} a_k, \max\limits_{1 \le k \le m} b_k)$.

7. **Use truth tables to prove the two-valued propositional tautologies:**

   (a) [A $\longrightarrow$ (B OR C)] $\longrightarrow$ [(A $\longrightarrow$ B) OR (A $\longrightarrow$ C)] ,

   (b) [A $\longrightarrow$ (B AND C)] $\longrightarrow$ [(A $\longrightarrow$ B) AND (A $\longrightarrow$ C)] ,

   (c) [(A OR B)$\longrightarrow$ C] $\longrightarrow$ [(A $\longrightarrow$ C) OR (B $\longrightarrow$ C)] ,

   (d) [(A $\longrightarrow$ C) AND (B $\longrightarrow$ C)] $\longrightarrow$ [(A AND B) $\longrightarrow$ C] .

   Is the converse of (c) a tautology? Explain whether this affects BIOFAM inference.

8. BIOFAM inference. Suppose the input spaces X and Y are both $[-10, 10]$, and the output space Z is $[-100, 100]$. Define five trapezoidal fuzzy sets–NL, NS, ZE, PS, PL— on X, Y, and Z. Suppose the underlying (unknown) system transfer function is $z = x^2 - y^2$. State at least five FAM rules that accurately describe the system's

behavior. Use $z = x^2 - y^2$ to generate streams of sample data. Use BIOFAM inference and fuzzy-centroid defuzzification to map input pairs (x, y) to output data z. Plot the BIOFAM outputs and the desired outputs z. What is the arithmetic average of the squared errors $(F(x, y) - x^2 + y^2)^2$? Divide the product space $X \times Y \times Z$ into 125 overlapping FAM cells. Estimate FAM rules from clustered system data (x, y, z). Use these FAM rules to control the system. Evaluate the performance.

Software Problems

The following problems use the accompanying FAM software for controlling an inverted pendulum.

1. Explain why the pendulum stabilizes in the diagonal position if the pendulum bob mass increases to maximum and the motor current decreases slightly. The pendulum stabilizes in the vertical position if you remove which FAM rules?

2. Oscillation results if you remove which FAM rules? The pendulum sticks in a horizontal equilibrium if you remove which FAM rules?

ADAPTIVE FUZZY SYSTEM FOR TARGET TRACKING

Peter J. Pacini and Bart Kosko
Department of Electrical Engineering
Signal and Image Processing Institute
University if Southern California
Los Angeles, CA 90089-0272

ABSTRACT

We compared fuzzy and Kalman-filter control systems for realtime target tracking. Both systems performed well, but in the presence of mild process (unmodeled effects) noise the fuzzy system exhibited finer control. We tested the robustness of the fuzzy controller by removing random subsets of fuzzy associations or "rules" and by adding destructive or "sabotage" fuzzy rules to the fuzzy system. We tested the robustness of the Kalman tracking system by increasing the variance of the unmodeled-effects noise process. The fuzzy controller performed well until we removed over 50% of the fuzzy rules. The Kalman controller's performance quickly degraded as the unmodeled-effects variance increased. We used unsupervised neural-network learning to adaptively generate the fuzzy controller's fuzzy-associative-memory structure. The fuzzy systems did not require a mathematical model of how system outputs depended on inputs.

Fuzzy and Math-Model Controllers

Fuzzy controllers differ from classical math-model controllers. Fuzzy controllers do not require a mathematical model of how control outputs functionally depend on control inputs. Fuzzy controllers also differ in the type of uncertainty they represent and how they represent it. The fuzzy approach represents ambiguous or fuzzy system behavior as partial implications or approximate "rules of thumb"—as fuzzy associations (A_i, B_i).

Fuzzy controllers are fuzzy systems. A finite *fuzzy set* A is a *point* [Kosko, 1987] in a unit hypercube $I^n = [0,1]^n$. A *fuzzy system* $F : I^n \rightarrow I^p$ is a *mapping* between unit hypercubes. I^n contains all fuzzy subsets of the domain space $X = \{x_1, \ldots, x_n\}$. I^n is the *fuzzy power set* $F(2^X)$ of X. I^p contains all the fuzzy subsets of the range space $Y = \{y_1, \ldots, y_p\}$. Element $x_i \in X$ belongs to fuzzy set A to degree $m_A(x_i)$. The 2^n nonfuzzy subsets of X correspond to the 2^n corners of the fuzzy cube I^n. The fuzzy system F maps fuzzy subsets of X to fuzzy subsets of Y. In general, X and Y are continuous not discrete sets.

Math-model controllers usually represent system uncertainty with probability distributions. Probability models describe system behavior with first-order and second-order statistics—with conditional means and covariances. They usually describe unmodeled effects and measurement imperfections with additive "noise" processes.

Mathematical models of the system state and measurement processes facilitate a mean-squared-error analysis of system behavior. In general we cannot accurately articulate such mathematical models. This greatly restricts the range of realworld applications. In practice we often use linear or quasi-linear (Markov) mathematical models.

Mathematical state and measurement models also make it difficult to add non-mathematical knowledge to the system. Experts may articulate such knowledge, or neural networks may adaptively infer it from sample data. In practice, once we have articulated the math model, we use human expertise only to estimate the initial state and covariance conditions.

Fuzzy controllers consist of a bank of *fuzzy associative memory* (FAM) "rules" or associations (A_i, B_i) operating in parallel, and operating to different degrees. Each FAM

rule is a set-level implication. It represents ambiguous expert knowledge or learned input-output transformations. A FAM rule can also summarize the behavior of a specific mathematical model. The system nonlinearly transforms exact or fuzzy state inputs to a fuzzy set output. This output fuzzy set is usually "defuzzified" with a centroid operation to generate an exact numerical output. In principle the system can use the entire fuzzy distribution as the output. We can easily construct, process, and modify the FAM bank of FAM rules in software or in digital VLSI circuitry.

Fuzzy controllers require that we articulate or estimate the FAM rules. The fuzzy-set framework provides more expressiveness than, say, traditional expert-system approaches, which encode bivalent propositional associations. But the fuzzy framework does not eliminate the burden of knowledge acquisition. We can use neural network systems to estimate the FAM rules. But neural systems also require an accurate (statistically representative) set of articulated input-output numerical samples. Below we use unsupervised competitive learning to adaptively generate target-tracking FAM rules.

Experts can hedge their system descriptions with fuzzy concepts. Although fuzzy controllers are numerical systems, experts can contribute their knowledge in natural language. This is especially important in complex problem domains, such as economics, medicine, and history, where we may not know how to mathematically model system behavior.

Below we compare a fuzzy controller with a Kalman-filter controller for realtime target tracking. This problem admits a simple and reasonably accurate mathematical description of its state and measurement processes. We chose the Kalman filter as a benchmark because of its many optimal linear-systems properties. We wanted to see whether this "optimal" controller remains optimal when compared with a computationally lighter fuzzy controller in different uncertainty environments.

We indirectly compared the sensitivity of the two controllers by varying their system uncertainties. We randomly removed FAM rules from the fuzzy controller. We also added "sabotage" FAM rules to the controller. Both techniques modeled less-stuctured control environments. For the Kalman filter, we varied the noise variance of the unmodeled-effects noise process.

Both systems performed well for mildly uncertain target environments. They degraded

differently as the system uncertainty increases. The fuzzy controller's performance degraded when we removed more than half the FAM rules. The Kalman-filter controller's performance quickly degraded when the additive state noise process increased in variance.

Realtime Target Tracking

A target tracking system maps azimuth-elevation inputs to motor control outputs. The nominal target moves through azimuth-elevation space. Two motors adjust the position of a platform to continuously point at the target.

The platform can be any directional device that accurately points at the target. The device may be a laser, video camera, or high-gain antenna. We assume we have available a radar or other device that can detect the direction from the platform to the target.

The radar sends azimuth and elevation coordinates to the tracking system at the end of each time interval. We calculate the current *error* e_k in platform position and *change in error* $\dot{e}_k$. Then a fuzzy or Kalman-filter controller determines the control outputs for the motors, one each for azimuth and elevation. The control outputs reposition the platform.

We can independently control movement along azimuth and elevation if we apply the same algorithm twice. This reduces the problem to matching the target's position and velocity in only one dimension.

Figure 1 shows a block diagram of the target tracking system. The controller's output v_k gives the estimated change in angle required during the next time interval. In principle a hardware system must transduce the angular velocity v_k into a voltage or current.

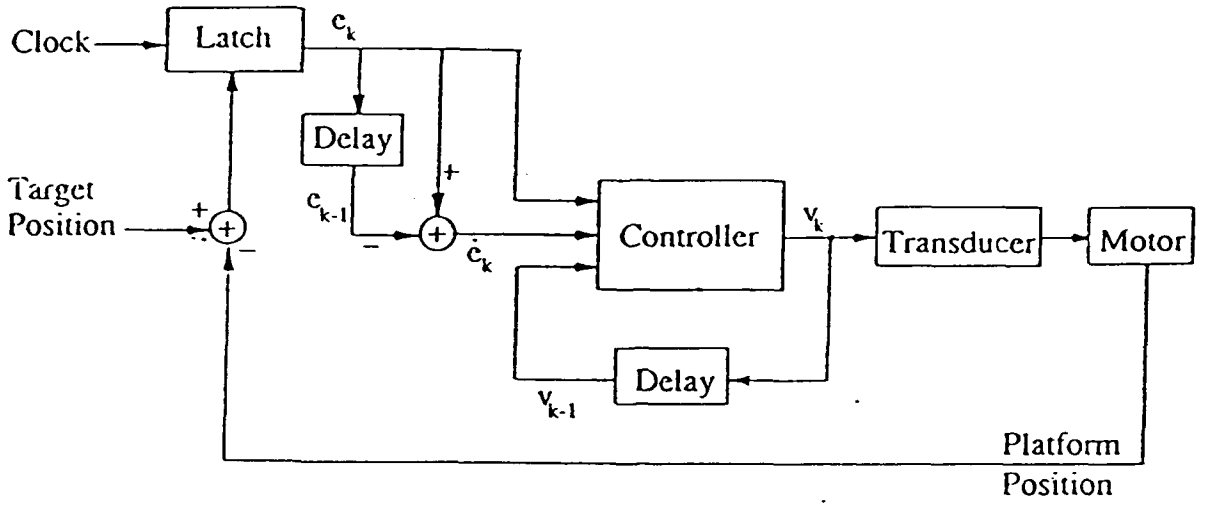


FIGURE 1 Target tracking system.

FUZZY CONTROLLER

We restrict the output angular velocity v_k of the fuzzy controller to the interval $[-6, 6]$. So we must insert a gain element before the voltage transduction. This gain must equal one-sixth the maximum angle through which the platform can turn in one time interval. Similarly, the position error e_k must be scaled so that 6 equals the maximum error. The product of this scale factor and the output gain provides a design parameter—the "gain" of the fuzzy controller.

The fuzzy controller uses heuristic control set-level "rules" or *fuzzy associative memory* (FAM) associations based on quantized values of e_k, $\dot{e}_k$, and v_{k-1}. We define seven fuzzy levels by the following library of fuzzy-set values of the fuzzy variables e_k, $\dot{e}_k$, and v_{k-1}:

LN : Large Negative

MN : Medium Negative

SN : Small Negative

ZE : Zero

SP : Small Positive

MP : Medium Positive

LP : Large Positive

We do not quantize inputs in the classical sense that we assign each input to exactly one output level. Instead, each linguistic value equals as a fuzzy set that overlaps with adjacent fuzzy sets. The fuzzy controller uses trapezoidal fuzzy-set values, as Figure 2 shows. The lengths of the upper and lower bases provide design parameters that we must calibrate for satisfactory performance. A good rule of thumb is ***adjacent fuzzy-set values should overlap approximately 25 percent.*** Below we discuss examples of calibrated and uncalibrated systems. The fuzzy controller attained its best performance with upper and lower bases of 1.2 and 3.9—26.2% overlap. Different target scenarios may require more or less overlap.

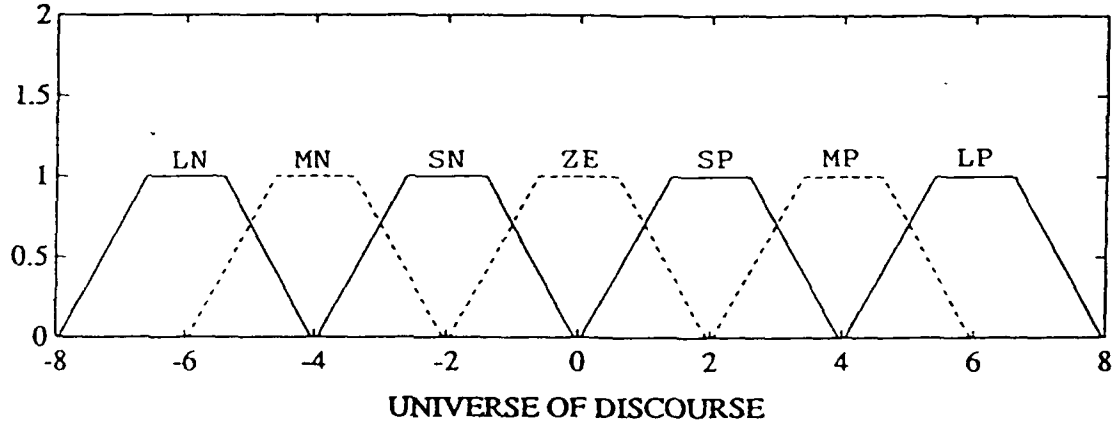


FIGURE 2 Library of overlapping fuzzy-set values defined on a universe

of discourse.

We assign each system input to a fit vector of length 7, where the ith *fit*, or *fuzzy unit* [Kosko, 1986], equals the value of the ith fuzzy set at the input value. In other words, the ith fit measures the degree to which the input belongs to the ith fuzzy-set value. For instance, we apply the input values $1, -4$, and 3.8 to the seven fuzzy sets in the library to obtain the fit vectors

$$\begin{aligned} 1 &\longrightarrow (0 \;\; 0 \;\; 0 \;\; .7 \;\; .7 \;\; 0 \;\; 0) \;\; , \\ -4 &\longrightarrow (0 \;\; 1 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0) \;\; , \\ 3.8 &\longrightarrow (0 \;\; 0 \;\; 0 \;\; 0 \;\; .1 \;\; 1 \;\; 0) \;\; . \end{aligned}$$

We determine these fit values above by convolving a Dirac delta function centered at the input value with each of the 7 fuzzy sets:

$$m_{SP}(3.8) \;=\; \delta(y \,-\, 3.8) \,*\, m_{SP}(y) \;=\; .1 \;\; . \tag{1}$$

If we use a discretized universe of discourse, then we use a Kronecker delta function instead. Equivalently, for the discrete case n-dimensional universe of discourse $X = \{x_1, \ldots, x_n\}$, a control input corresponds to a *bit* (binary unit) vector B of length n. A single 1 element in the ith slot represents the "crisp" input value x_i. Similarly, we represent the kth library fuzzy set by an n-dimensional fit vector A_k that contains samples of the fuzzy set at the n discrete points within the universe of discourse X. The degree to which the crisp input x_i activates each fuzzy set equals the inner product $B \cdot A_k$ of the bit vector B and the corresponding fit vector A_k.

We formulate control FAM rules by associating output fuzzy sets with input fuzzy sets. The antecedent of each FAM rule conjoins e_k, $\dot{e}_k$, and v_{k-1} fuzzy-set values. For example,

$$\text{IF } e_k = MP \text{ AND } \dot{e}_k = SN \text{ AND } v_{k-1} = ZE, \text{ THEN } v_k = SP.$$

We abbreviate this as $(MP, SN, ZE; SP)$.

The scalar activation value w_i of the ith FAM rule's consequent equals the *minimum* of the three antecedent conjuncts' values. If alternatively we combine the antecedents disjunctively with OR, the activation degree of the consequent would equal the *maximum* of the three antecedent disjuncts' values. In the following example, $m_A(e_k)$ denotes the degree to which e_k belongs to the fuzzy set A:

				LN	MN	SN	ZE	SP	MP	LP
e_k	=	2.6	$\longrightarrow$	(0	0	0	0	1	.4	0)
$\dot{e}_k$	=	-2.0	$\longrightarrow$	(0	0	1	0	0	0	0)
v_{k-1}	=	1.8	$\longrightarrow$	(0	0	0	.1	1	0	0)
$m_{MP}(e_k)$	=	.4								
$m_{SN}(\dot{e}_k)$	=	1								
$m_{ZE}(v_{k-1})$	=	.1								
w_i	=	min(.4, 1, .1)	=	.1	.					

So the system activates the consequent fuzzy set SP to degree $w_i = .1$.

The output fuzzy set's shape depends on the FAM-rule encoding scheme used. With *correlation-minimum* encoding, we clip the consequent fuzzy set L_i in the library of fuzzy-set values to degree w_i with pointwise minimum:

$$m_{O_i}(y) = \min(w_i, m_{L_i}(y)) \ . \tag{2}$$

With *correlation-product* encoding, we multiply L_i by w_i:

$$m_{O_i}(y) = w_i \, m_{L_i}(y) \ , \tag{3}$$

or equivalently,

$$O_i = w_i \, L_i \ . \tag{4}$$

Figure 3 illustrates how both inference procedures transform L_i to scaled output O_i. For

the example above, correlation-product inference gives output fuzzy set $O_i = .1SP$, where $L_i = SP$ denotes the fuzzy set of small but positive angular velocity values.

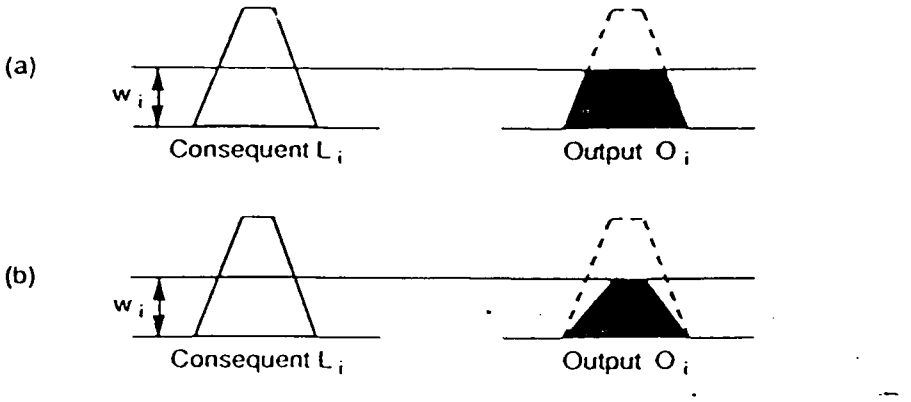


FIGURE 3 FAM inference procedure depends on FAM rule encoding procedure: (a) correlation-minimum encoding, (b) correlation-product encoding.

The fuzzy system activates each FAM rule consequent set to a different degree. For the ith FAM rule this yields the output fuzzy set O_i. The system then sums the O_i to form the combined output fuzzy set O:

$$O = \sum_{i=1}^{N} O_i \ , \tag{5}$$

or equivalently,

$$m_O(y) = \sum_{i=1}^{N} m_{O_i}(y) \ . \tag{6}$$

The control output v_k equals the *fuzzy centroid* of O:

$$v_k = \frac{\int y \, m_O(y) dy}{\int m_O(y) dy} \ , \tag{7}$$

where the limits of integration correspond to the entire universe of discourse Y of angular velocity values. Figure 4 shows an example of correlation-product inference for two FAM rules followed by centroid defuzzification of the combined output fuzzy set.

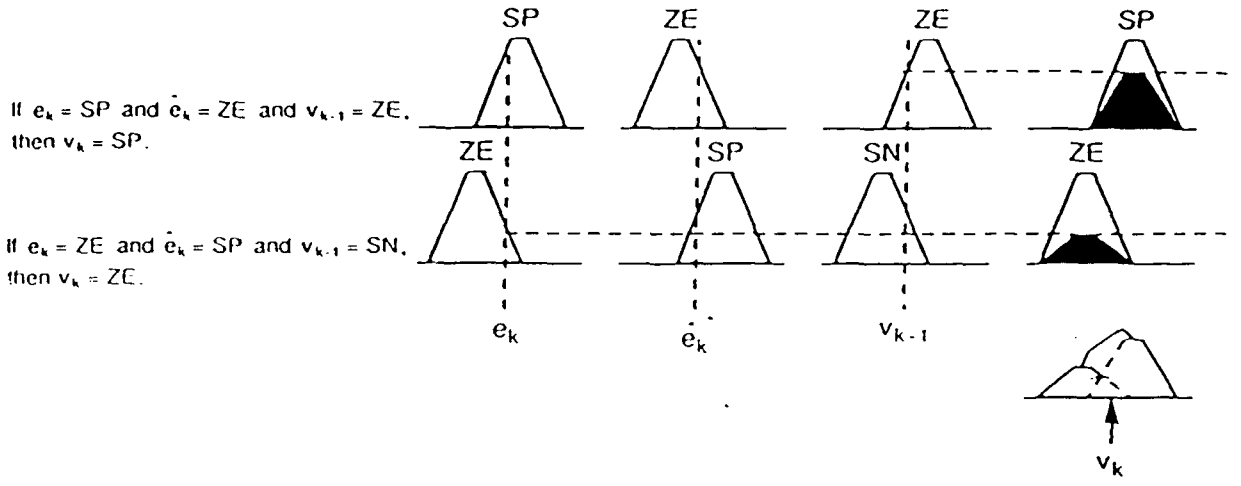


FIGURE 4 Correlation-product inferences followed by centroid defuzzification. FAM rule antecedents combined with AND use the *minimum* fit value to activate consequents. Those combined with OR use the *maximum* fit value.

To reduce computations, we can discretize the output universe of discourse Y to p values, $Y = \{y_1, \ldots, y_p\}$, which gives the discrete fuzzy centroid

$$v_k = \frac{\sum_{j=1}^{p} y_j \, m_O(y_j)}{\sum_{j=1}^{p} m_O(y_j)} \quad . \tag{8}$$

Fuzzy Centroid Computation

We now develop two discrete methods for computing the fuzzy centroid (7). Theorem 1 states that we can compute the global centroid v_k from local FAM-rule centroids. Theorem 2 states that v_k can be computed from only 7 sample points if all the fuzzy sets

are symmetric and unimodal (in the broad sense of a trapezoid peak), though otherwise arbitrary. Both results reduce computation and favor digital implementation.

Theorem 1: If correlation-product inference determines the output fuzzy sets, then we can compute the global centroid v_k from local FAM-rule centroids:

$$v_j = \frac{\sum_{i=1}^{N} w_i c_i I_i}{\sum_{i=1}^{N} w_i I_i} \quad . \tag{9}$$

Proof. The consequent fuzzy set of each FAM rule equals one of the fuzzy-set values shown in Figure 2. We assume each fuzzy set includes at least one unity value, $m_A(x) = 1$. Define I_i and c_i as the respective area and centroid of the ith FAM rule's consequent set L_i:

$$I_i = \int m_{L_i}(y)dy \quad , \tag{10}$$

$$c_i = \frac{\int y\, m_{L_i}(y)dy}{\int m_{L_i}(y)dy}$$

$$= \frac{\int y\, m_{L_i}(y)dy}{I_i} \quad ,$$

substituting from (10). Hence

$$\int y\, m_{L_i}(y)dy = c_i\, I_i \quad . \tag{11}$$

Using (3), the result of correlation-product inference, we get

$$\int y\, m_{O_i}(y)dy = \int y\, w_i\, m_{L_i}(y)dy$$

$$= w_i \int y\, m_{L_i}(y)dy$$

$$= w_i c_i I_i \quad , \tag{12}$$

substituting from (11). Similarly,

$$\int m_{O_i}(y)\, dy = \int w_i\, m_{L_i}(y)dy$$

$$= w_i I_i \quad , \tag{13}$$

substituting from (10).

We can use (12) and (13) to derive a discrete expression equivalent to (7):

$$\int y\, m_O(y)dy = \int y[\sum_{i=1}^{N} m_{O_i}(y)]dy \quad \text{substituting from (6)} \quad ,$$

$$= \sum_i \int y\, m_{O_i}(y)\, dy$$

$$= \sum_i w_i c_i I_i \quad , \tag{14}$$

from (12). Similarly,

$$\int m_O(y)dy = \int \sum_{i=1}^{N} m_{O_i}(y)dy$$

$$= \sum_i \int m_{O_i}(y)dy$$

$$= \sum_i w_i I_i \quad , \tag{15}$$

from (13). Substituting (14) and (15) into (7), we derive a new form for the centroid:

$$v_k = \frac{\sum_{i=1}^{N} w_i c_i I_i}{\sum_{i=1}^{N} w_i I_i} \quad , \tag{16}$$

which is equivalent to (9). Each summand in each summation of (16) depends on only a single FAM rule. So we can compute the global output centroid from local FAM-rule centroids. Q.E.D.

Theorem 2: If the 7 library fuzzy sets are symmetric and unimodal (in the trapezoidal sense) and we use correlation-product inference, then we can compute the centroid v_k from only 7 samples of the combined output fuzzy set O:

$$v_k = \frac{\sum_{j=1}^{7} m_O(y_j)\ y_j\ J_j}{\sum_{j=1}^{7} m_O(y_j)\ J_j} \quad . \tag{17}$$

The 7 sample points are the centroids of the output fuzzy-set values.

Proof. Define $\bar{O}_i$ as a fit vector of length 7, where the fit value corresponding to the ith consequent set has the value w_i, and the other entries equal zero. If all the fuzzy sets are symmetric and unimodal, then the jth fit value of $\bar{O}_i$ is a sample of m_{O_i} at the centroid of the jth fuzzy set. The combined output fit vector is

$$\bar{O} = \sum_{i=1}^{N} \bar{O}_i \quad . \tag{18}$$

Since

$$m_O(y) = \sum_{i=1}^{N} m_{O_i}(y) \quad ,$$

the jth fit value of $\bar{O}$ is a sample of m_O at the centroid of the jth fuzzy set. Equivalently, the jth fit value of $\bar{O}$ equals the sum of the output activations w_i from the FAM rules with

consequent fuzzy sets equal to the jth library fuzzy-set value.

Define the reduced universe of discourse as $Y = \{y_1, \ldots, y_7\}$ such that y_j equals the centroid of the jth output fuzzy set. In vector form

$$\begin{aligned} Y &= (y_1, \ldots, y_7) \\ &= (-6, \ -4, \ -2, \ 0, \ 2, \ 4, \ 6) \end{aligned}$$

for the library of fuzzy sets in Figure 2. Also define the diagonal matrix

$$J = \text{diag}(J_1, \ldots, J_7) \quad , \tag{19}$$

where J_j denotes the area of the jth fuzzy-set value. If the ith FAM rule's consequent fuzzy set equals the jth fuzzy-set value, then the jth fit value of $\bar{O}$ increases by w_i, $c_i = y_j$, and $I_i = J_j$. So

$$\bar{O}JY^T = \sum_{j=1}^{7} m_O(y_j) y_j J_j = \sum_{i=1}^{N} w_i c_i I_i \quad . \tag{20}$$

Also,

$$\bar{O}J\mathbf{1}^T = \sum_{j=1}^{7} m_O(y_j) J_j = \sum_{i=1}^{N} w_i I_i \quad , \tag{21}$$

where $\mathbf{1} = (1, \ldots, 1)$. Substituting (20) and (21) into (16) gives

$$v_k = \frac{\sum_{j=1}^{7} m_O(y_j) \ y_j \ J_j}{\sum_{j=1}^{7} m_O(y_j) \ J_j} \quad , \tag{22}$$

which is equivalent to (17). Therefore, (22) gives a simpler, but equivalent form of the centroid (7) if all the fuzzy sets are symmetric and unimodal, and if we use correlation-product inference to form the output fuzzy sets O_i. Q.E.D.

Consider a fuzzy controller with the fuzzy sets defined in Figure 2, and 7 FAM rules with the following outputs:

i	w_i	Consequent
1	0.0	MP
2	0.2	SP
3	1.0	ZE
4	0.4	SN
5	0.1	SP
6	0.8	ZE
7	0.6	SN

Figure 5 shows the combined output fuzzy set O, with the SN, ZE, and SP components displayed with dotted lines. Using (7) we get a velocity output of -0.452. Alternatively, the combined output fit vector $\bar{O}$ equals (0, 0, 1.0, 1.8, 0.3, 0, 0). From (22) we get

$$v_k = \frac{-2 \times 1 + 0 \times 1.8 + 2 \times 0.3}{1 + 1.8 + 0.3} = -0.452 \quad .$$

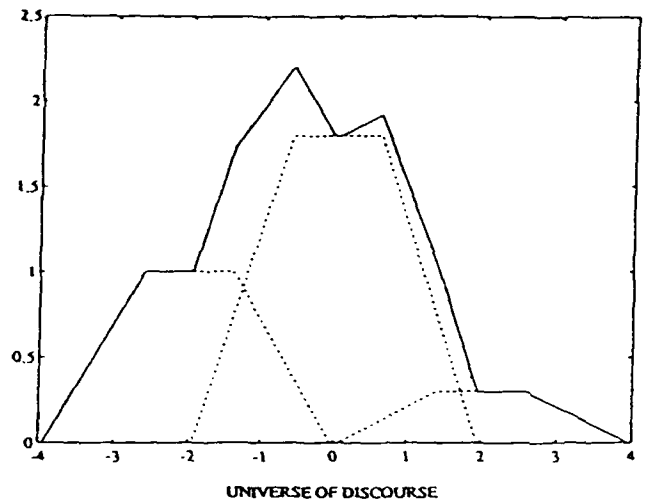


FIGURE 5 Output fuzzy set O.

Fuzzy Controller Implementation

A FAM bank or "rulebase" of FAM rules defines the fuzzy controller. Each FAM rule associates one consequent fuzzy set with three antecedent fuzzy-set conjuncts.

Suppose the ith FAM rule is $(MP, SN, ZE; SP)$. Suppose the inputs at time k are $e_k = 2.6$, $\dot{e}_k = -2.0$, and $v_{k-1} = 1.8$. Then

$$\begin{aligned} w_i &= \min(m_{MP}(e_k),\ m_{SN}(\dot{e}_k),\ m_{ZE}(v_{k-1})) \\ &= \min(.4,\ 1,\ .1) \\ &= .1\ . \end{aligned}$$

If all the fuzzy sets have the same shape, then they correspond to shifted versions of a

single fuzzy set ZE:

$$m_{SP}(y) \;=\; m_{ze}(y-2) \quad .$$

Define e^i, $\dot{e}^i$, and v^i as the centroids of the corresponding antecedent fuzzy sets in the example above. So $e^i = 4$, $\dot{e}^i = -2$, and $v^i = 0$. Then the output activation equals

$$\begin{aligned} w_i &= \min(m_{ZE}(e_k - e^i),\ m_{ZE}(\dot{e}_k - \dot{e}^i),\ m_{ZE}(v_{k-1} - v^i)) \\ &= \min(m_{ZE}(-1.4),\ m_{ZE}(0),\ m_{ZE}(1.8)) \\ &= \min(.4,\ 1,\ .1) \\ &= .1 \ , \end{aligned}$$

as computed above. Figure 6 schematizes such a FAM rule when presented with crisp inputs.

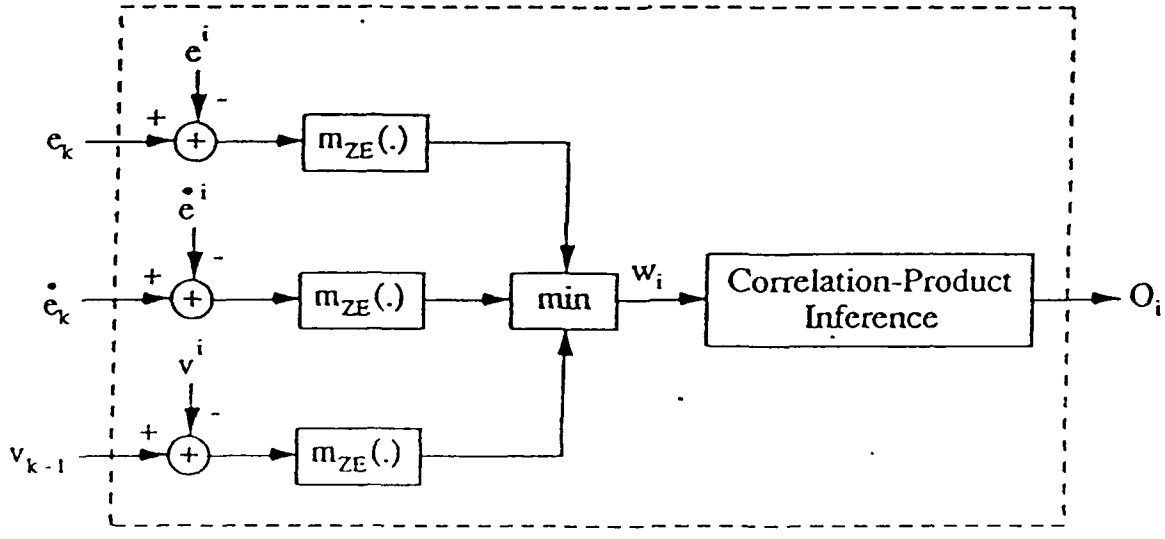


FIGURE 6 Algorithmic structure of a FAM rule for the special case of identically-shaped fuzzy sets and correlation-product inference.

The output fuzzy set O_i in Figure 6 equals the fuzzy set ZE scaled by w_i and shifted by c_i:

$$m_{O_i}(y) = w_i m_{ZE}(y - c_i) \quad . \tag{23}$$

Figure 7 illustrates O_i.

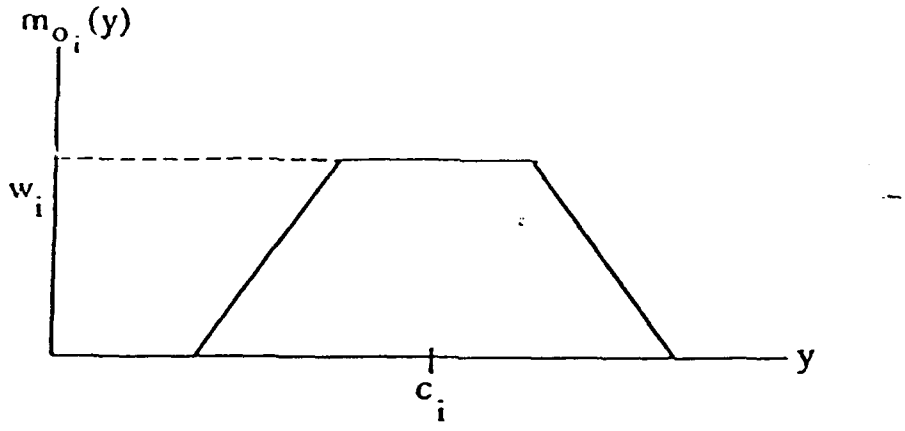


FIGURE 7 Trapezoidal output fuzzy set O_i.

The fuzzy control system activates a bank of FAM rules operated in parallel, as shown in Figure 8. The system sums the output fuzzy sets to form the total output set O, which the system converts to a "defuzzified" scalar output by computing its fuzzy centroid.

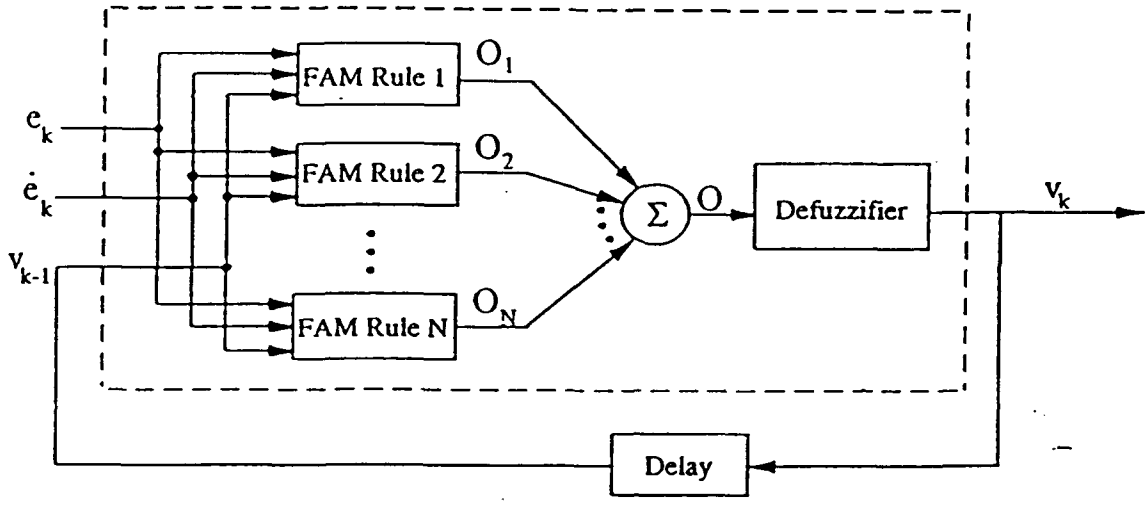


FIGURE 8 Fuzzy control system as a parallel FAM bank with centroidal output.

KALMAN FILTER CONTROLLER

We designed a one-dimensional **Kalman filter** to act as an alternative controller. The *state* and *measurement* equations take the general form

$$\begin{aligned} x_{k+1} &= \Phi_{k+1,k}\, x_k + \Gamma_{k+1,k}\, w_k + \Psi_{k+1,k}\, u_k \quad , \\ z_k &= H_k\, x_k + V_k \quad , \end{aligned} \tag{24}$$

where V_k denotes Gaussian white noise with covariance matrix R_k. If V_k is colored noise or if $R_k = 0$, then the filtering-error covariance matrix $P_{k|k}$ becomes singular. The state x_k and the measurements z_k are jointly Gaussian. Mendel [1987] gives details of this model.

Assume the following one-dimensional model:

$$\Phi_{k+1,k} = \Gamma_{k+1,k} = \Psi_{k+1,k} = H_k = 1 \text{ for all } k,$$

$$u_k = e_k + \dot{e}_k \quad . \tag{25}$$

Let x_{k+1} denote the output velocity required at time k to exactly lock onto the target at time $k+1$. So the controller output at time k equals the "predictive" estimate $\hat{x}_{k+1|k} = v_k$. Note that

$$\begin{aligned} e_k &= x_k - \hat{x}_{k|k-1} \\ &= \tilde{x}_{k|k-1} \ , \\ \dot{e}_k &= e_k - e_{k-1} \ . \end{aligned}$$

Substituting (25) into (24), we get the new state equation

$$x_{k+1} = x_k + e_k + \dot{e}_k + w_k \ , \tag{26}$$

where w_k denotes white noise that models target acceleration or other unmodeled effects. The new measurement equation is

$$\begin{aligned} z_k &= x_k + V_k \\ &= \hat{x}_{k|k-1} + \tilde{x}_{k|k-1} + V_k \\ &= \hat{x}_{k|k-1} + V_k' \ . \end{aligned} \tag{27}$$

Since we assume $\tilde{x}_{k|k-1}$ and V_k are uncorrelated, the variance of V_k' is

$$\begin{aligned} R_k' &= E[V_k'^2] \\ &= E[\tilde{x}_{k|k-1}^2] + E[V_k^2] \end{aligned} \tag{28}$$

$$= P_{k|k-1} + R_k \quad .$$

The general form of the recursive Kalman filter equations is

$$\begin{aligned}
\hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k[z_k - H_k \hat{x}_{k|k-1}] \quad , \\
K_k &= P_{k|k-1} H_k^T [H_k P_{k|k-1} H_k^T + R_k]^{-1} \quad , \\
\hat{x}_{k+1|k} &= \Phi_{k+1,k} \hat{x}_{k|k} + \Psi_{k+1,k} u_k \quad , \\
P_{k|k-1} &= \Phi_{k,k-1} P_{k-1|k-1} \Phi_{k,k-1}^T + \Gamma_{k,k-1} Q_{k-1} \Gamma_{k,k-1}^T \quad , \\
P_{k|k} &= [I - K_k H_k] P_{k|k-1} \quad ,
\end{aligned} \tag{29}$$

where $Q_k = Var(w_k) = E[w_k w_k^T]$. Substituting (25), (27), (28) and the definition of v_k into (29), we get the following one-dimensional Kalman filter:

$$\begin{aligned}
\hat{x}_{k|k} &= v_{k-1} + K_k V_k' \quad , \\
K_k &= \frac{P_{k|k-1}}{R_k'} \quad , \\
v_k &= \hat{x}_{k|k} + e_k + \dot{e}_k \quad , \\
P_{k|k-1} &= P_{k-1|k-1} + Q_{k-1} \quad , \\
P_{k|k} &= [1 - K_k] P_{k|k-1} \quad .
\end{aligned} \tag{30}$$

Unlike the fuzzy controller, this Kalman filter does not automatically restrict the output v_k to a usable range. We must apply a threshold immediately after the controller. To remain consistent with the fuzzy controller, we set the following thresholds:

$$\begin{aligned}
|v_k| &\leq 9 \text{ degrees azimuth} \quad , \\
|v_k| &\leq 4.5 \text{ degrees elevation.}
\end{aligned}$$

Fuzzy and Kalman Filter Control Surfaces

Each control system maps inputs to outputs. Geometrically, these input-output transformations define *control surfaces*. The control surfaces are sheets in the input space (since the output velocity v_k is a scalar). Three inputs and one output give rise to a four-dimensional control surface, which we cannot plot. Instead, for each controller we can plot a family of three-dimensional control surfaces indexed by constant values of the fourth variable, the error e_k, say. Then each control surface corresponds to a different value of the error e_k.

The fuzzy control surface characterizes the fuzzy system's fuzzy-set value definitions and its bank of FAM rules. Different sets of FAM rules yield different fuzzy controllers, and hence different control surfaces. Figure 9 shows a cross section of the FAM bank when $e_k = ZE$. Each entry in this linguistic matrix represents one FAM rule with $e_k = ZE$ as the first antecedent term.

$\dot{e}_k$ \ v_{k-1}	LN	MN	SN	ZE	SP	MP	LP
LN	LN	LN	LN	LN	MN	SN	ZE
MN	LN	LN	LN	MN	SN	ZE	SP
SN	LN	LN	MN	SN	ZE	SP	MP
ZE	LN	MN	SN	ZE	SP	MP	LP
SP	MN	SN	ZE	SP	MP	LP	LP
MP	SN	ZE	SP	MP	LP	LP	LP
LP	ZE	SP	MP	LP	LP	LP	LP

FIGURE 9 $e_k = ZE$ cross section of the fuzzy control system's FAM bank. Each entry represents one FAM rule with $e_k = ZE$ as the first antecedent term.

The shaded FAM rule is "IF $e_k = ZE$ AND $\dot{e}_k = SP$ AND $v_{k-1} = SN$, THEN $v_k = ZE$," abbreviated as $(ZE, SP, SN; ZE)$. Note the ordinal anti-symmetry of this FAM-bank matrix. The six other cross-section FAM-bank matrices are similar. We can eliminate many FAM rule entries without greatly perturbing the fuzzy controller's behavior.

The entire FAM bank—including cross sections for e_k equal to each of the seven fuzzy-set values LN, MN, SN, ZE, SP, MP, and LP—determines how the system maps input fuzzy sets to output fuzzy sets. The fuzzy set membership functions shown in Figure 2 determine the degree to which each crisp input value belongs to each fuzzy-set value. So both the fuzzy-set value definitions and the FAM bank determine the defuzzified output v_k for any set of crisp input values e_k, $\dot{e}_k$, and v_{k-1}.

Figure 10 shows the control surface of the fuzzy controller for $e_k = 0$. We plotted the control output v_k against $\dot{e}_k$ and v_{k-1}. Since we use the same algorithm for tracking in azimuth and elevation, the control surfaces for the two dimensions differ in scale only by a factor of two.

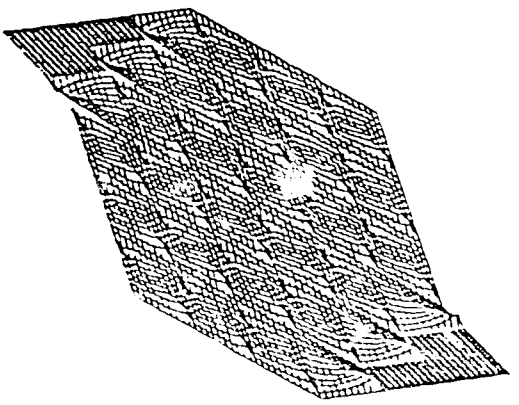


FIGURE 10 Control surface of the fuzzy controller for constant error $e_k = 0$. We plotted the control output v_k against $\dot{e}_k$ and v_{k-1} along the respective west and south borders.

The Kalman filter has a random control surface that depends on a time-varying pa-

rameter. From (30) we see that

$$v_k = \hat{x}_{k|k} + e_k + \dot{e}_k ,$$

$$\hat{x}_{k|k} = v_{k-1} + K_k V_k' ,$$

where V_k' denotes white noise with variance given by (28). Combining these two equations gives the equation for the random control surface:

$$v_k = v_{k-1} + e_k + \dot{e}_k + K_k V_k' . \tag{31}$$

At time k the noise term $K_k V_k'$ has variance

$$\sigma_k^2 = K_k^2 R_k' \tag{32}$$

$$= \frac{P_{k|k-1}^2}{R_k'} \text{ upon substituting from (30)} ,$$

$$= \frac{P_{k|k-1}^2}{P_{k|k-1} + R_k},$$

substituting from (28). Combining (31) and (32) gives a new control surface equation:

$$v_k = v_{k-1} + e_k + \dot{e}_k + \sigma_k V_k'' , \tag{33}$$

where V_k'' denotes unit-variance Gaussian noise. So the Kalman filter's control output equals the sum of the three input variables plus additive Gaussian noise with time-dependent variance σ_k^2. For constant error e_k, we can interpret (33) as a smooth control surface in R^3 defined by

$$v_k = v_{k-1} + e_k + \dot{e}_k ,$$

and perturbed at time k by Gaussian noise with variance σ_k^2.

In our simulations the standard deviation σ_k converged after only a few iterations. We used unity initial conditions: $P_{o|o} = R_k = 1$ for all k.

Table 1 lists the convergence rates and steady-state values of σ_k for three different values of the variance $Var(w)$ of the white-noise, unmodeled-effects process w_k. For $Var(w) = 0$, σ_k decreases rapidly at first—$\sigma_8 = .10$, $\sigma_{17} = .05$—but does not attain its steady-state value of zero within 100 iterations.

$Var(w)$	Steady-state value of σ_k	Number of iterations required for convergence
1.00	0.79	2
0.25	0.46	4
0.05	0.22	9

TABLE 1 Convergence rates and steady-state values of σ_k for different values of the variance $Var(w)$ of the white-noise, unmodeled-effects process w_k.

Figure 11 shows four realizations of the Kalman filter's random control surface for $c_k = 0$, each at a time k when σ_k has converged to its steady-state value. For each plot, we used output thresholds and initial variances for the azimuth case: $|v_k| \leq 9.0$, $R_k = P_{o|o} = 1.0$. As with the fuzzy controller, elevation control surfaces equal scaled versions of the corresponding azimuth control surfaces.

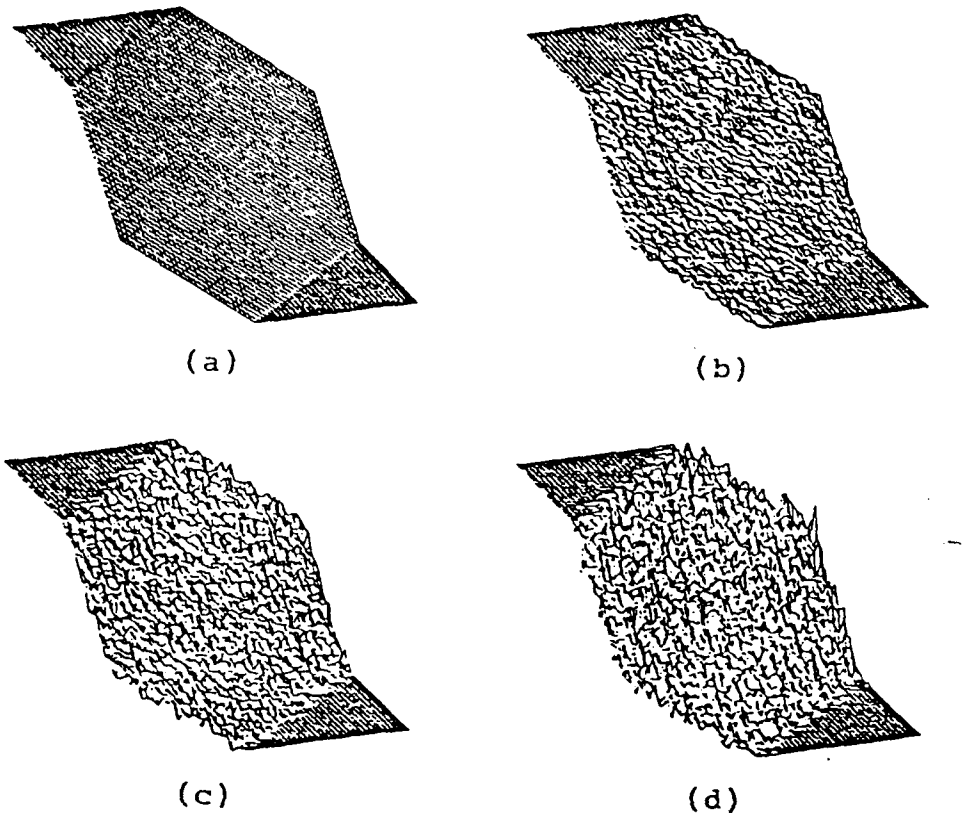


FIGURE 11 Realizations of the Kalman filter's random control surface with $e_k = 0$ for different values of the variance $Var(w)$ and steady-state values of the standard deviation σ_k: (a) $Var(w) = \sigma_k = 0$, (b) $Var(w) = .05$, $\sigma_k = .22$; (c) $Var(w) = .25$, $\sigma_k = .46$; (d) $Var(w) = 1.0$, $\sigma_k = .79$.

SIMULATION RESULTS

Our target-tracking simulations model several realworld scenarios. Suppose we have mounted the target tracking system on the side of a vehicle, aircraft, or ship. The system tracks a missile that cuts across the detection range on a straight flight path. The target maintains a constant speed of 1,870 miles-per-hour and comes within 3.5 miles of the

platform at closest approach. The platform can scan from 0 to 180 degrees in azimuth at a maximum rate of 36 degrees-per-second, and from 0 (vertical) to 90 degrees in elevation at a maximum rate of 18 degrees-per-second. The sampling interval is 1/4 of a second. The gain of the fuzzy controller equals 0.9. So the maximum error considered is 10 degrees azimuth and 5 degrees elevation. We threshold all error values above this level.

Figure 12 demonstrates the best performance of the fuzzy controller for a simulated scenario. The solid lines indicate target position. The dotted lines indicate platform position. To achieve this performance, we calibrated the three design parameters—upper and lower trapezoid bases and the gain. Figures 13 and 14 show examples of uncalibrated systems. Too much overlap causes excessive overshoot. Too little overlap causes lead or lag for several consecutive time intervals. A gain of 0.9 suffices for most scenarios. We can fine-tune the fuzzy control system by altering the percentage overlap between adjacent fuzzy sets.

Figure 15 demonstrates the best performance of the Kalman-filter controller for the same scenario used to test the fuzzy controller. For simplicity, $R_k = P_{0|0}$ for all values of k. For this study we chose the values 1.0 (unit variance) for azimuth and 0.25 for elevation. This 1/4 ratio reflects the difference in scanning range. We set Q_k to 0 for optimal performance. Figure 16 shows the Kalman-filter controller's performance when $Q_k = 1.0$ azimuth, 0.25 elevation.

Sensitivity Analysis

We compared the uncertainty sensitivity of the fuzzy and Kalman-filter control systems. Under normal operating conditions, when the FAM bank contains all fuzzy control rules, and when the unmodeled-effects noise variance $Var(w)$ is small, the controllers perform almost identically. Under more uncertain conditions their performance differs. The Kalman filter's state equation (26) contains the noise term w_k whose variance we must assume. When $Var(w)$ increases, the state equation becomes more uncertain. The fuzzy control

FAM rules depend implicitly on this same equation, but without the noise term. Instead, the fuzziness of the FAM rules accounts for the system uncertainty. This suggests that we can increase the uncertainty of the implicit state equation by omitting randomly selected FAM rules. Figures 17 and 18 show the effect on the *root-mean-squared error* (RMSE) in degrees when we omit FAM rules and increase $Var(w)$. Each data point averages ten runs.

The controllers behave differently as uncertainty increases. The RMSE of the fuzzy controller increases little until we omit nearly sixty percent of the FAM rules. The RMSE of the Kalman filter increases steeply for small values of $Var(w)$, then gradually levels off.

We also tested the fuzzy controller's robustness by "sabotaging" the most vulnerable FAM rule. This could reflect lack of accurate expertise, or a highly unstructured problem. Changing the consequent of the steady-state FAM rule $(ZE, ZE, ZE; ZE)$ to LP gives the following nonsensical FAM rule:

IF the platform points directly at the target
    AND both the target and the platform are stationary,
THEN turn in the positive direction with maximum velocity.

Figure 19 shows the fuzzy system's performance when this sabotage FAM rule replaces the steady-state FAM rule. When the sabotage FAM rule activates, the system quickly adjusts to decrease the error again. The fuzzy system is piecewise stable.

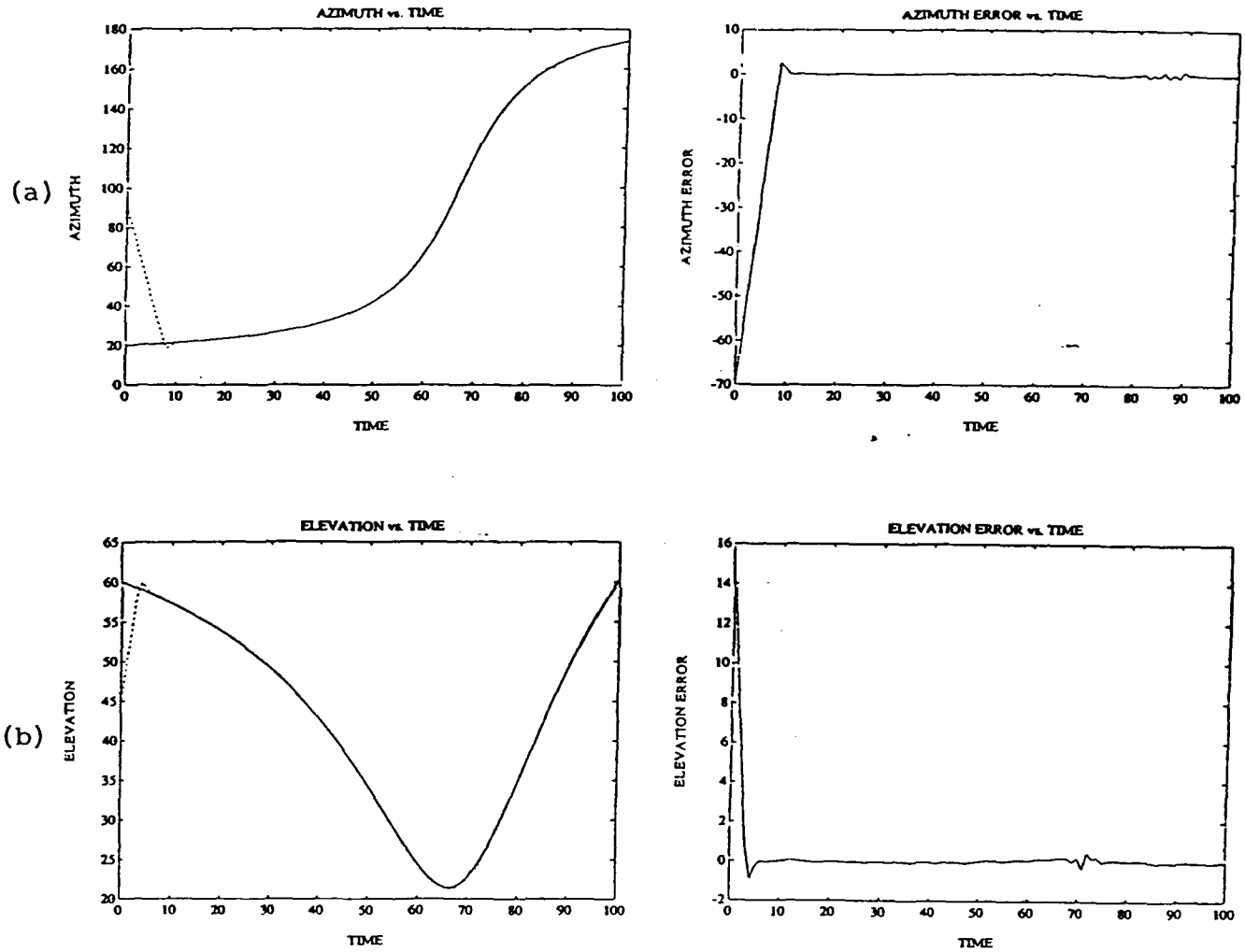


FIGURE 12 Best performance of the fuzzy controller: (a) azimuth position and error, (b) elevation position and error. Fuzzy set overlap is 26.2%.

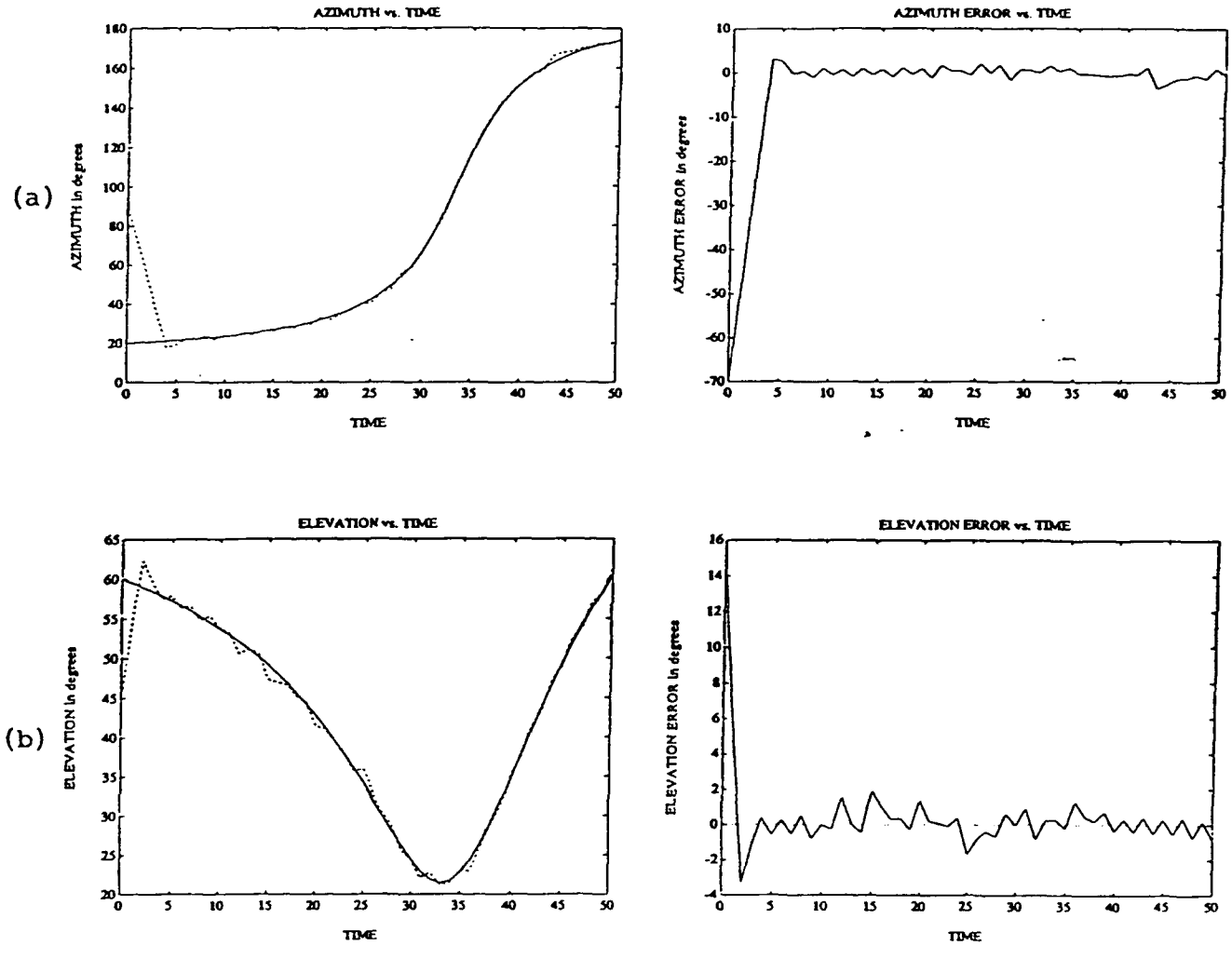


FIGURE 13 Uncalibrated fuzzy controller: (a) azimuth position and error, (b) elevation position and error. Fuzzy set overlap equals 33.3%. Too much overlap causes excessive overshoot.

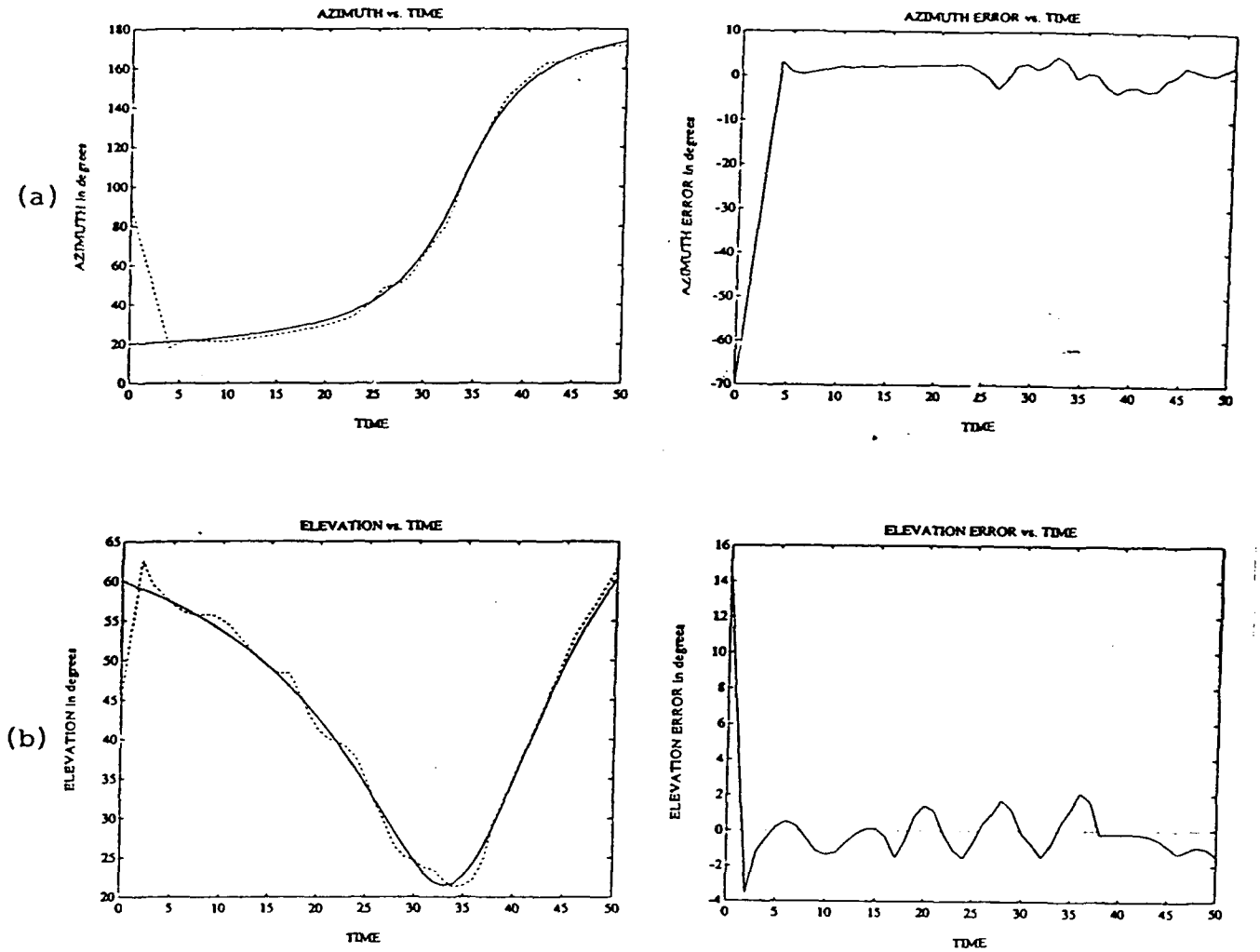


FIGURE 14 Uncalibrated fuzzy controller: (a) azimuth position and error, (b) elevation position and error. Fuzzy set overlap equals 12.5%. Too little overlap causes lead or lag for several consecutive time intervals.

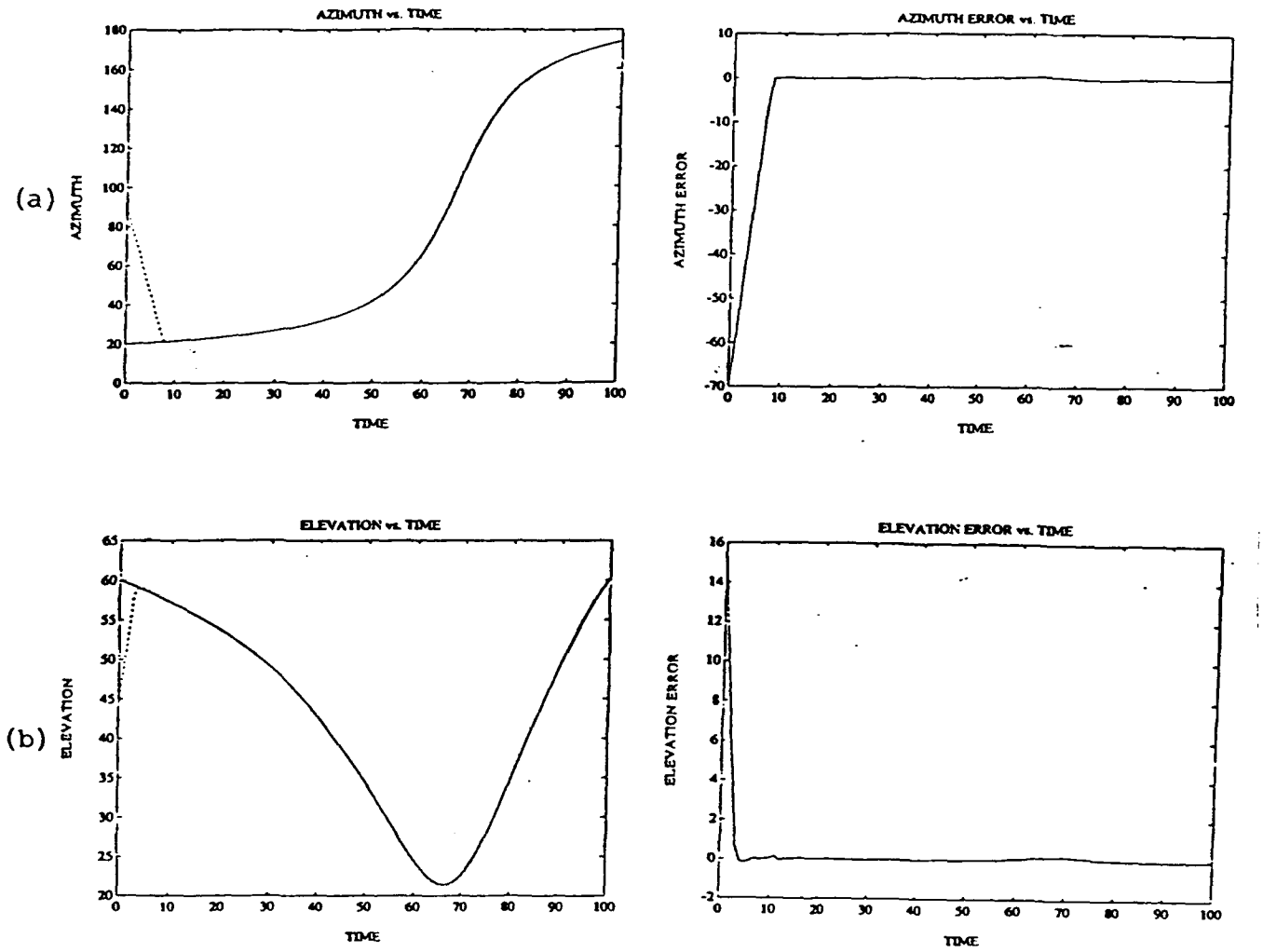


FIGURE 15 Kalman filter controller with unmodeled-effects noise variance $Var(w) = 0$: (a) azimuth position and error, (b) elevation position and error.

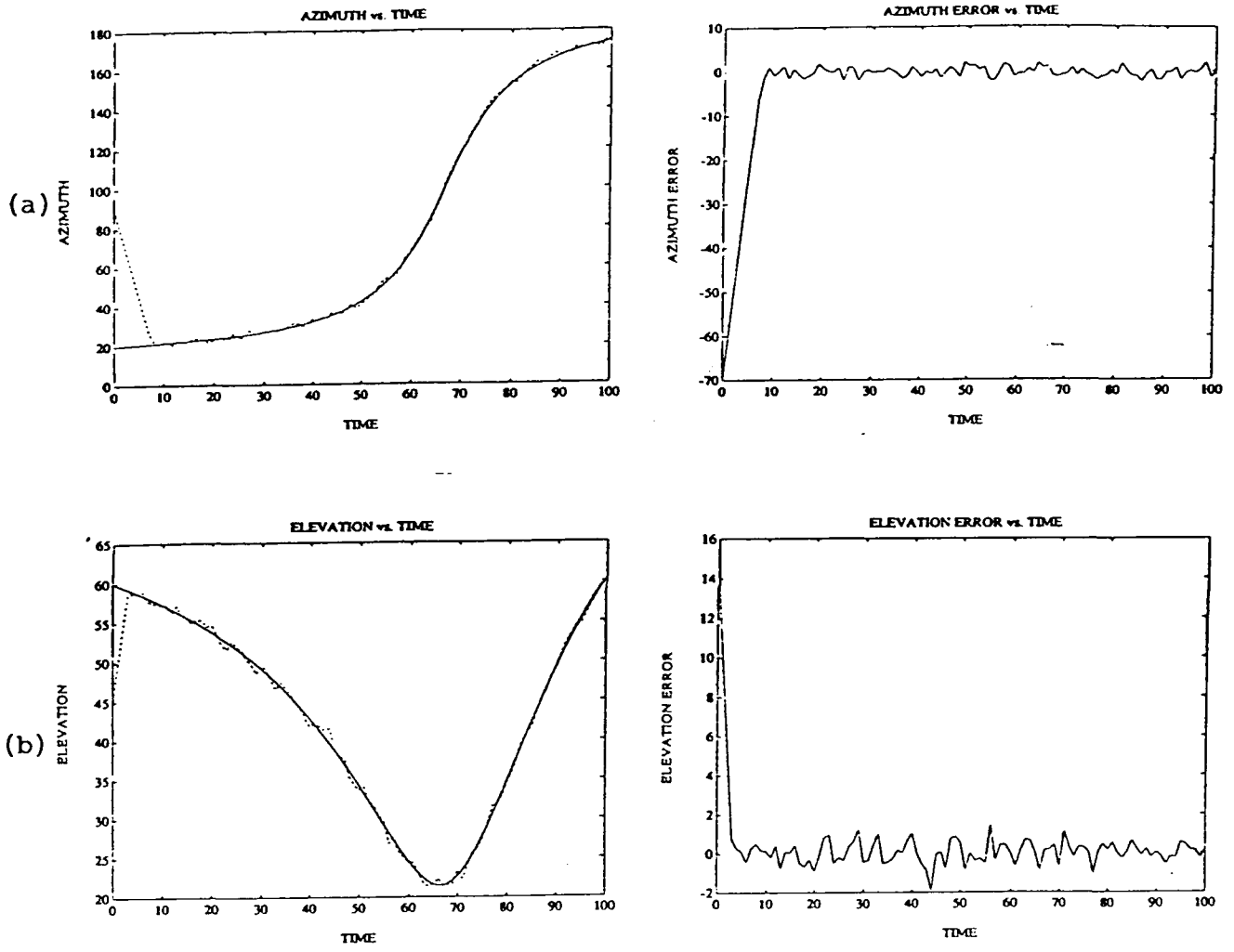


FIGURE 16 Kalman filter controller with $Var(w) = 1.0$ azimuth, 0.25 elevation: (a) azimuth position and error, (b) elevation position and error.

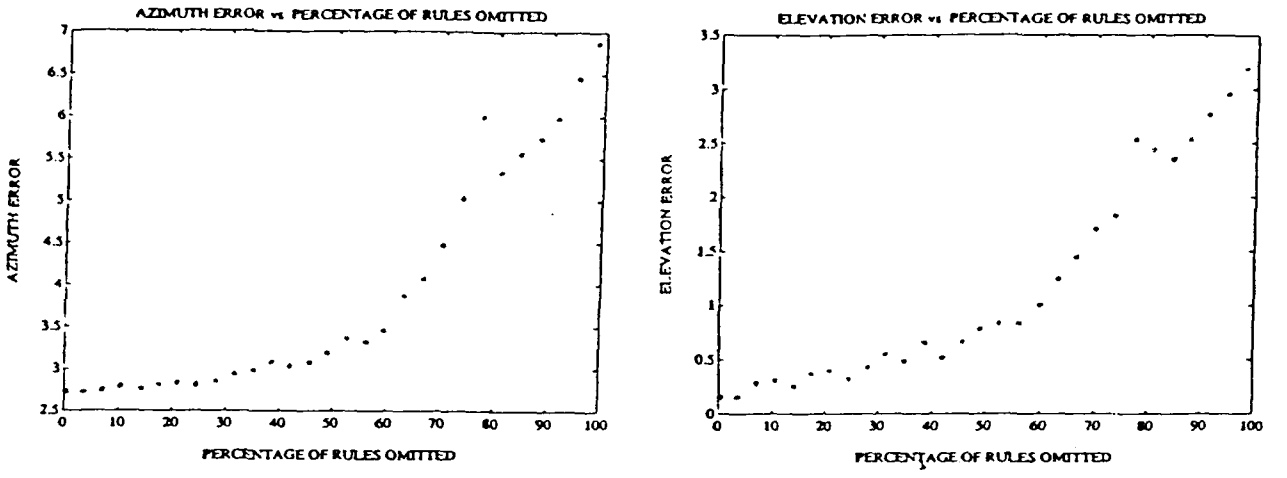


FIGURE 17 Root-mean-squared error of the fuzzy controller with randomly selected FAM rules omitted.

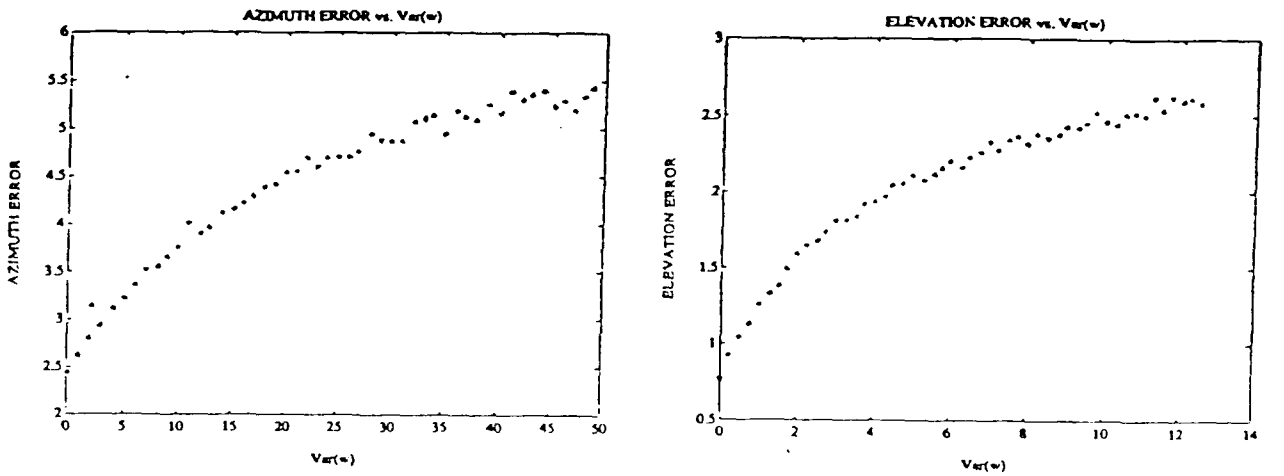


FIGURE 18 Root-mean-squared error of the Kalman filter controller as $Var(w)$ varies.

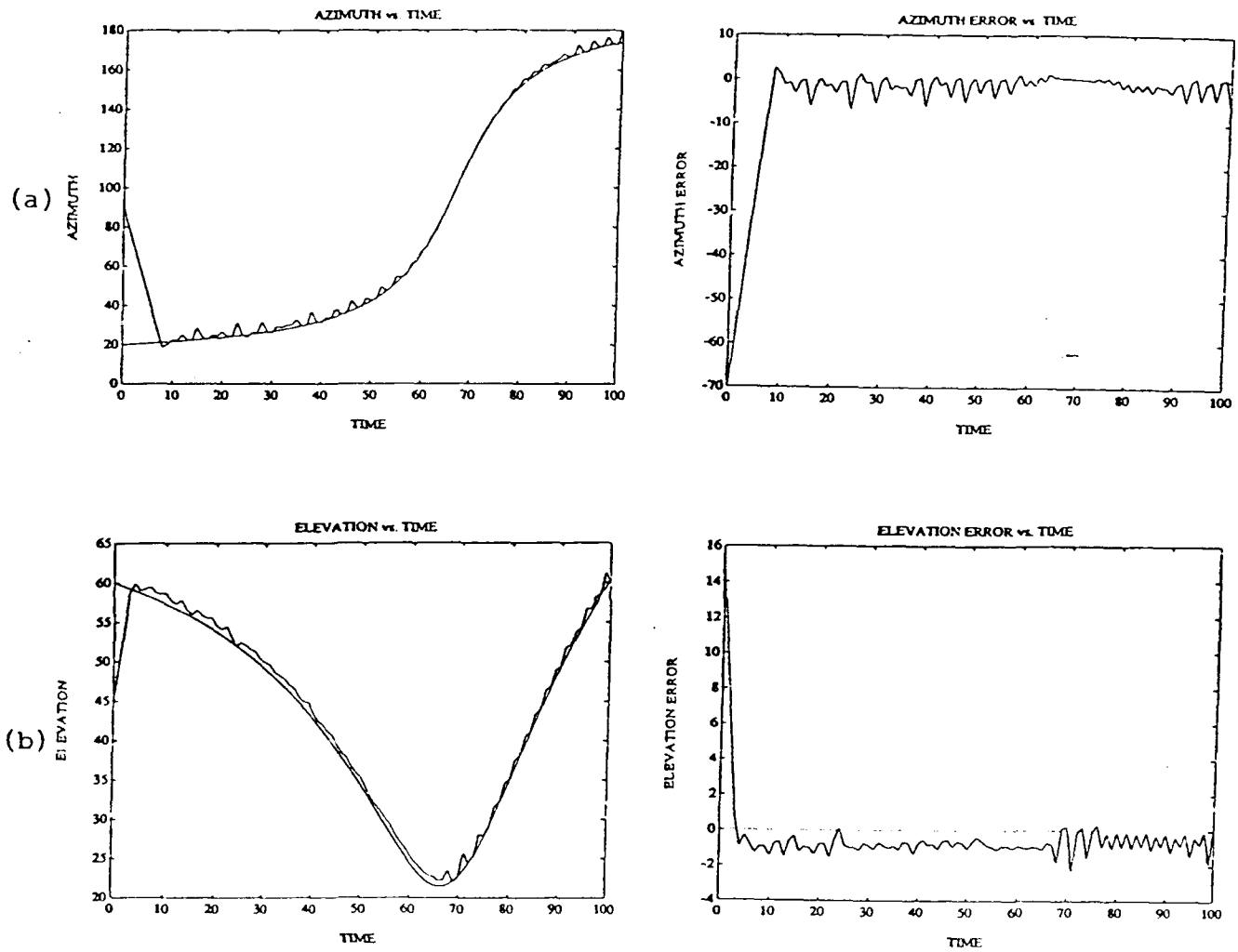


FIGURE 19 Fuzzy controller with a "sabotage" FAM rule: (a) azimuth position and error, (b) elevation position and error. The sabotage rule $(ZE, ZE, ZE; LP)$ replaces the steady-state FAM rule $(ZE, ZE, ZE; ZE)$. The system quickly adjusts each time the sabotage rule activates.

Adaptive FAM (AFAM)

We used unsupervised product-space clustering [Kosko, 1990a] to train an adaptive FAM (AFAM) fuzzy controller. Differential competitive learning (DCL) adaptively clustered input-output pairs. The Appendix describes product-space clustering with DCL. For this study, there were four input neurons in F_x. A manually-designed FAM bank and 80 random target trajectories generated 19,236 training vectors. Each product-space training vector $(e_k,\ \dot{e}_k,\ v_{k-1},\ v_k)$ defined a point in R^4.

Symmetry allowed us to reflect about the origin all sample vectors with negative errors e_k. We then trained 3,000 synaptic quantization vectors ($p = 3,000$) in the positive error half-space. For each sample vector, we defined the 10 closest synaptic vectors as "winners" ($N = 10$). The matrix W of F_Y within-field synaptic connection strengths had diagonal elements $w_{ii} = 2.9$, off-diagonal elements $w_{ij} = -0.1$. After training, we reflected the 3,000 synaptic quantization vectors about the origin to give 6,000 trained synaptic vectors.

The product-space FAM cells uniformly partitioned the four-dimensional product space. Each FAM cell represented a single FAM rule. The four fuzzy variables could assume only the 7 fuzzy-set values LN, MN, SN, ZE, SP, MP, and LP. So the product space contained $7^4 = 2401$ FAM cells.

At the end of the DCL training period, we defined a FAM cell as occupied only if it contained at least one synaptic vector. For some combinations of antecedent fuzzy sets, synaptic vectors occupied more than one FAM cell with different consequent fuzzy sets. In these cases we computed the centroid of the consequent fuzzy sets weighted by the number of synaptic vectors in their FAM cells. We chose the consequent fuzzy set as that output fuzzy-set value with centroid nearest the weighted centroid value. We ignored other FAM rules with the same antecedents but different consequent fuzzy sets.

Figure 20(a) shows the $e_k = ZE$ cross section of the original FAM bank used to generate the training samples. Figure 20(b) shows the same cross section of the DCL-estimated FAM bank. Figure 21 shows the original and DCL-estimated control surfaces for constant error $e_k = 0$.

for constant error $e_k = 0$.

The regions where the two control surfaces differ correspond to infrequent high-velocity situations. So the original and DCL-estimated control surfaces yield similar results. Table 2 compares the controllers' root-mean-squared errors for 10 randomly-selected target trajectories.

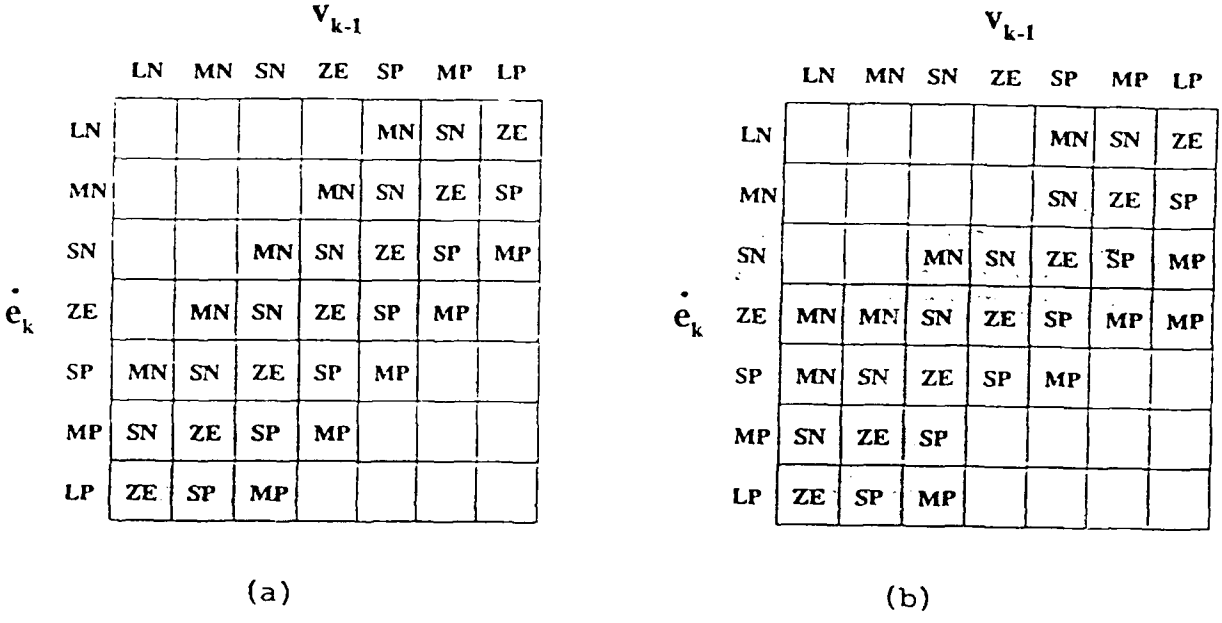
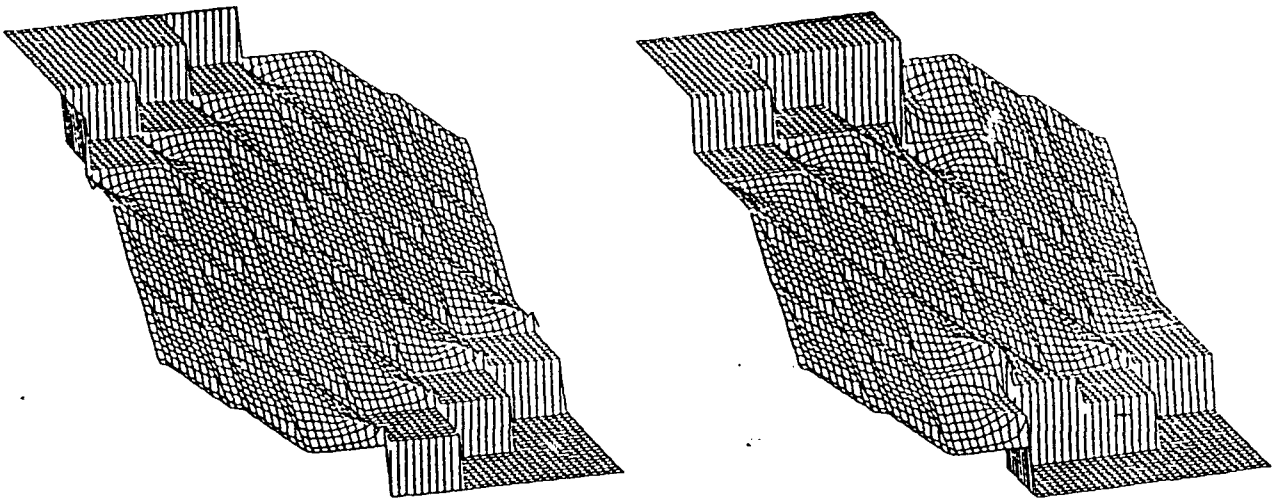


FIGURE 20 Cross sections of the original and DCL- estimated FAM banks when $e_k = ZE$: (a) original, (b) DCL- estimated.



(a) (b)

FIGURE 21 Control surfaces for constant error $e_k = 0$: (a) original, (b) DCL-estimated.

Trajectory	*Azimuth*		*Elevation*	
Number	Original	Estimated	Original	Estimated
1	2.33	2.33	3.31	3.37
2	4.14	4.14	3.03	2.89
3	6.11	6.11	3.69	3.68
4	3.83	3.83	3.32	3.30
5	4.02	4.02	3.11	3.10
6	2.84	2.84	1.20	1.21
7	3.22	3.22	3.04	2.98
8	0.75	0.74	2.00	2.00
9	9.28	9.27	5.50	5.41
10	1.81	1.81	2.29	2.29
Average	3.83	3.83	3.05	3.02

TABLE 2 Root-mean-squared errors for 10 randomly-selected target trajectories. The original and DCL-estimated FAM banks yielded similar results since they differed only in regions corresponding to infrequent high-velocity situations.

Conclusion

We developed and compared a fuzzy control system and a Kalman-filter control system for realtime target tracking. The fuzzy system represented uncertainty with continuous or fuzzy sets, with the partial occurence of multiple alternatives. The Kalman-filter system represented uncertainty with the random occurence of an exact alternative. Accordingly, our simulations tested each system's response to a different family of uncertainty environments, one fuzzy and the other random. In general representative training data can "blindly" generate the governing FAM rules.

These simulations suggest that in many cases fuzzy controllers may be a robust, computationally effective alternative to linear Kalman filter, indeed to nonlinear extended Kalman filter, approaches to realtime system control—even when we can accurately articulate an input-output math model.

REFERENCES

Huber, P.J., *Robust Statistics*, Wiley, 1981.

Kong, S.-G., Kosko, B., "Differential Competitive Learning for Centroid Estimation and Phoneme Recognition," *IEEE Transactions of Neural Networks*, to appear, January 1991.

Kosko, B., "Fuzzy Entropy and Conditioning," *Information Sciences*, vol.40, 165-174, 1986.

Kosko, B., *Foundations of Fuzzy Estimation Theory*, Ph.D. dissertation, Department of Electrical Engineering, University of California at Irvine, June 1987; Order number 8801936, University Microfilms International, 300 N. Zeeb Road, Ann Arbor, Michigan 48106.

Kosko, B., *Neural Networks and Fuzzy Systems: A Dynamical System Approach to Machine Intelligence*, Prentice-Hall, 1990.

Kosko, B., "Stochastic Competitive Learning," *Proceedings of the Summer 1990 International Joint Conference on Neural Networks (IJCNN-90)*, vol.II, 215-226, June 1990.

Kosko, B., "Unsupervised Learning in Noise," *IEEE Transactions on Neural Networks*, vol.1, no.1, 44-57, March 1990.

Mendel, J.M., *Lessons in Digital Estimation Theory*, Prentice-Hall, 1987.

Acknowledgment

This research was supported by the Air Force Office of Scientific Research (AFOSR-88-0236) and by a grant from the Rockwell Science Center.

Appendix: Product-space Clustering with Differential Competitive Learning

Adaptive Vector Quantization

Product-space clustering [Kosko, 1990a] is a form of stochastic adaptive vector quantization. Adaptive vector quantization (AVQ) systems adaptively quantize pattern clusters in R^n. Stochastic competitive-learning systems are neural AVQ systems. Neurons compete for the activation induced by randomly sampled patterns. The corresponding fan-in vectors adaptively quantize the pattern space R^n. The p synaptic vectors $\mathbf{m}_j$ define the p columns of the synaptic connection matrix M. M interconnects the n input or linear neurons in the input neuronal field F_X to the p competing nonlinear neurons in the output field F_Y. Figure 22 below illustrates the neural network topology.

Learning algorithms estimate the unknown probability density function $p(\mathbf{x})$, which describes the distribution of patterns in R^n. More synaptic vectors arrive at more probable regions. Where sample vectors $\mathbf{x}$ are dense or sparse, synaptic vectors $\mathbf{m}_j$ should be dense or sparse. The local count of synaptic vectors then gives a nonparametric estimate of the volume density $P(V)$ for volume $V \subset R^n$:

$$P(V) = \int_V p(\mathbf{x})d\mathbf{x} \tag{34}$$

$$\approx \frac{\text{Number of } \mathbf{m}_j \in V}{p} \quad . \tag{35}$$

In the extreme case that $V = R^n$, this approximation gives $P(V) = p/p = 1$. For improbable subsets V, $P(V) = 0/p = 0$.

Stochastic Competitive Learning Algorithms

The metaphor of competing neurons reduces to nearest-neighbor classification. The AVQ system compares the current vector random sample $\mathbf{x}(t)$ in Euclidean distance to the p columns of the synaptic connection matrix M, to the p synaptic vectors $\mathbf{m}_1(t), \ldots, \mathbf{m}_p(t)$. If the jth synaptic vector $\mathbf{m}_j(t)$ is closest to $\mathbf{x}(t)$, then the jth output neuron "wins" the competition for activation at time t. In practice we sometimes define the nearest N synaptic vectors as winners. Some scaled form of $\mathbf{x}(t) - \mathbf{m}_j(t)$ updates the nearest or "winning" synaptic vectors. "Losers" remain unchanged: $\mathbf{m}_i(t+1) = \mathbf{m}_i(t)$. Competitive synaptic vectors converge to pattern-class centroids exponentially fast [Kosko, 1990b].

The following three-step process describes the competitive AVQ algorithm, where the third step depends on which learning algorithm updates the winning synaptic vectors.

Competitive AVQ Algorithm

1. Initialize synaptic vectors $\mathbf{m}_i(0) = \mathbf{x}(i), i = 1, \ldots, p$. Sample-dependent initialization avoids many pathologies that can distort nearest-neighbor learning.

2. For random sample $\mathbf{x}(t)$, find the closest or "winning" synaptic vector $\mathbf{m}_j(t)$:

$$\|\mathbf{m}_j(t) - \mathbf{x}(t)\| = \min_i \|\mathbf{m}_i(t) - \mathbf{x}(t)\| \quad , \tag{36}$$

where $\|\mathbf{x}\|^2 = x_1^2 + \ldots + x_n^2$ defines the squared Euclidean vector norm of $\mathbf{x}$. We can define the N synaptic vectors closest to $\mathbf{x}$ as "winners."

3. Update the winning synaptic vector(s) $\mathbf{m}_j(t)$ with an appropriate learning algorithm.

Differential Competitive Learning (DCL)

Differential competitive "synapses" learn only if the competing "neuron" changes its competitive status [Kosko, 1990c]:

$$\dot{m}_{ij} = \dot{S}_j(y_j)[S_i(x_i) - m_{ij}] \quad , \tag{37}$$

or in vector notation,

$$\dot{\mathbf{m}}_j = \dot{S}_j(y_j)[\mathbf{S}(\mathbf{x}) - \mathbf{m}_j] \quad , \tag{38}$$

where $\mathbf{S}(\mathbf{x}) = (S_1(x_1), \ldots, S_n(x_n))$ and $\mathbf{m}_j = (m_{1j}, \ldots, m_{nj})$. m_{ij} denotes the synaptic value between the ith neuron in input field F_X and the jth neuron in competitive field F_Y. Nonnegative signal functions S_i and S_j transduce the real-valued activations x_i and y_j into bounded monotone nondecreasing signals $S_i(x_i)$ and $S_j(y_j)$. $\dot{m}_{ij}$ and $\dot{S}_j(y_j)$ denote the time derivatives of m_{ij} and $S_j(y_j)$, synaptic and signal velocities. $S_j(y_j)$ measures the competitive status of the jth competing neuron in F_Y. Usually S_j approximates a binary threshold function. For example, S_j may equal a steep binary logistic sigmoid,

$$S_j(y_j) = \frac{1}{1 + e^{-cy_j}} \quad , \tag{39}$$

for some constant $c > 0$. The jth neuron wins the laterally inhibitive competition if $S_j = 1$, loses if $S_j = 0$.

For discrete implementation, we use the DCL algorithm as a stochastic difference equation [Kong, 1991]:

$$\mathbf{m}_j(t+1) = \mathbf{m}_j(t) + c_t \, \Delta S_j(y_j(t))[S(\mathbf{x}(t)) - \mathbf{m}_j(t)] \text{ if the } j\text{th neuron wins,} \tag{40}$$

$$\mathbf{m}_i(t+1) \;=\; \mathbf{m}_i(t) \;\text{ if the } i\text{th neuron loses.} \tag{41}$$

$\Delta S_j(y_j(t))$ denotes the time change of the jth neuron's competition signal $S_j(y_j)$ in the competition layer F_Y:

$$\Delta S_j(y_j(t)) \;=\; sgn[S_j(y_j(t+1)) \;-\; S_j(y_j(t))] \quad . \tag{42}$$

We define the signum operator $sgn(x)$ as

$$sgn(x) \;=\; \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases} \quad . \tag{43}$$

$\{c_t\}$ denotes a slowly decreasing sequence of learning coefficients, such as $c_t = .1(1 - t/2000)$ for 2000 training samples. Stochastic approximation [Huber, 1981] requires a decreasing gain sequence $\{c_t\}$ to suppress random disturbances and to guarantee convergence to local minima of mean-squared performance measures. The learning coefficients should decrease slowly,

$$\sum_{t=1}^{\infty} c_t \;=\; \infty \quad , \tag{44}$$

but not too slowly,

$$\sum_{t=1}^{\infty} c_t^2 \;<\; \infty \quad . \tag{45}$$

Harmonic-series coefficients, $c_t = 1/t$, satisfy these constraints.

We approximate the competitive signal difference ΔS_j as the activation difference Δy_j:

$$\Delta S_j(y_j(t)) \;=\; sgn[y_j(t+1) \;-\; y_j(t)] \tag{46}$$

$$\;=\; \Delta y_j(t) \quad . \tag{47}$$

Input neurons in feedforward networks usually behave linearly: $S_i(x_i) = x_i$, or $S(\mathbf{x}(t)) = \mathbf{x}(t)$. Then we update the winning synaptic vector $\mathbf{m}_j(t)$ with

$$\mathbf{m}_j(t+1) = \mathbf{m}_j(t) + c_t \, \Delta y_j(t)[\mathbf{x}(t) - \mathbf{m}_j(t)] \quad . \tag{48}$$

We update the F_Y neuronal activations y_j with the additive model

$$y_j(t+1) = y_j(t) + \sum_i^n S_i(x_i(t))m_{ij}(t) + \sum_k^p S_k(y_k(t))w_{kj} \quad . \tag{49}$$

For linear signal functions S_i, the first sum in (49) reduces to an inner product of sample and synaptic vectors:

$$\sum_i^n x_i(t)m_{ij}(t) = \mathbf{x}^T(t)\mathbf{m}_j(t) \quad . \tag{50}$$

Then positive learning tends to occur—$\Delta m_{ij} > 0$—when $\mathbf{x}$ is close to the jth synaptic vector $\mathbf{m}_j$.

Since a binary threshold function approximates the output signal function $S_k(y_k)$, the second sum in (49) sums over just the winning neurons: $\sum_k w_{kj}$ for all winning neurons y_k .

The $p \times p$ matrix W contains the F_Y within-field synaptic connection strengths. Diagonal elements w_{ii} are positive, off-diagonal elements negative. Winning neurons excite themselves and inhibit all other neurons. Figure 22 shows the connection topology of the laterally inhibitive DCL network.

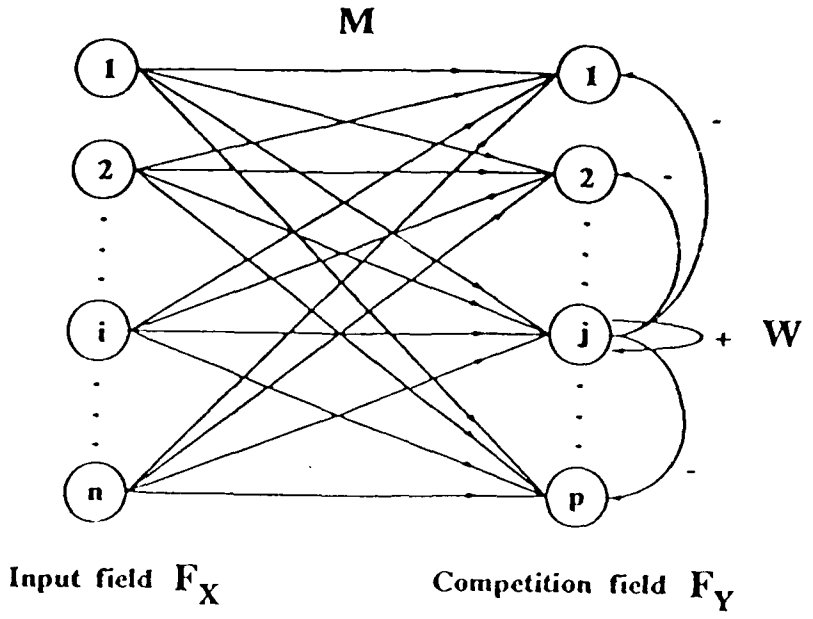


FIGURE 22 Topology of the laterally inhibitive DCL network.

In: June IJCNN-90 Proceedings

STOCHASTIC COMPETITIVE LEARNING

Bart Kosko
Department of Electrical Engineering
Signal and Image Processing Institute
University of Southern California
Los Angeles, California 90089-0272

Abstract

The probabilistic foundations of competitive learning systems are developed. Continuous and discrete formulations of unsupervised, supervised, and differential competitive learning systems are studied. These systems estimate an unknown probability density function from random pattern samples and behave as adaptive vector quantizers. Synaptic vectors, in feedforward competitive neural networks, quantize the pattern space and converge to pattern class centroids or local probability maxima. The stochastic calculus and a Lyapunov argument prove that competitive synaptic vectors converge to centroids exponentially quickly. Convergence does not depend on a specific dynamical model of how neuronal activations change.

Feedforward Multilayer Competitive Learning Systems

Competitive learning systems are usually feedforward multilayer neural networks. Neurons compete for the activation induced by randomly sampled pattern vectors $\mathbf{x} \in R^n$. An unknown probability density function $p(\mathbf{x})$ characterizes the continuous distribution of random pattern vectors $\mathbf{x}$. A random n-vector $\mathbf{m}_j$ of synaptic values fans-in to each competing neuron. The synaptic vector $\mathbf{m}_j$ is the jth column of the synaptic connection matrix M.

Competition selects which synaptic vector $\mathbf{m}_j$ is modified by the training sample $\mathbf{x}$. In practice competition is a metaphor for metrical pattern matching. Neuronal activation dynamics are seldom used. The jth neuron "wins" at an iteration if the synaptic vector $\mathbf{m}_j$ is the closest, in Euclidean distance, of the m synaptic vectors to the random pattern $\mathbf{x}$ sampled at that iteration.

Some scaled form of the difference vector $\mathbf{x} - \mathbf{m}_j$ additively modifies the closest synaptic vector $\mathbf{m}_j$. Different scaling factors determine different competitive learning systems. A positive scaling factor "rewards" the winning jth neuron by making the modified synaptic vector $\mathbf{m}_j$ resemble the random sample $\mathbf{x}$ at least as much as did the unmodified synaptic vector $\mathbf{m}_j$. A negative scaling factor "punishes" the jth neuron by making the modified synaptic vector $\mathbf{m}_j$ disresemble $\mathbf{x}$ more than did the unmodified synaptic vector. Geometrically, negative scaling tends to move misclassifying synaptic vectors in R^n out of regions of misclassification.

Competitive learning systems can be viewed as non-neural signal processing algorithms. As in a correlation detection system, random inputs are metrically compared with the columns of the matrix M. Pattern recognition (signal detection) is nearest-neighbor template matching. During training - and the system may always be training - at most one column of M is modified at a time. This lightens computation and favors digital implementation.

Autoassociative[11] competitive learning systems have two layers or fields of neurons. The input data field F_X of n neurons passes a randomly sampled pattern vector $\mathbf{x}$ forward through an n-by-m matrix M of synaptic values to m "competing" neurons in the competitive field F_Y. A symmetric m-by-m matrix W of within-field synaptic values describes the competition in F_Y. W has a positive main diagonal (or diagonal band) with negative or zero values elsewhere. In practice W has 1s down its main diagonal and -1s elsewhere. Autoassociative competitive learning systems recognize patterns. More generally, they estimate probability density functions $p(\mathbf{x})$.

Heteroassociative[7] competitive learning systems have three fields of neurons. The first and third fields are input and output data fields for sampling the random vector association $(\mathbf{x}, \mathbf{z})$. The second or "hidden" field contains the competing neurons. The first and third fields can be concatenated into a single field of

$n + p$ neurons. Competitive learning proceeds as in the autoassociative case. Closest matches modify the columns of the between-field matrices M and N of synaptic values with the same difference-based learning law.

In practice heteroassociative competitive learning systems do not directly estimate an unknown joint probability density function $p(\mathbf{x}, \mathbf{z})$. Instead they estimate a sampled continuous function $f : R^n \rightarrow R^p$ from a large number of noisy random vector samples $(\mathbf{x_i}, \mathbf{z_i})$. Implicitly the functional pair $(\mathbf{x}_i, f(\mathbf{x}_i))$ belongs to a high-probability region of $R^n \times R^p$. These heteroassociative systems can be directly compared with multilayer neural networks trained with the backpropagation algorithm and with the same training data. The competitive systems tend to learn faster, but less accurately, at least for small dimensions n and p.

Feedback autoassociative and heteroassociative competitive learning systems, such as random adaptive bidirectional associative memories[16], have more complicated dynamics. These stochastic systems are globally stable if within-field competition is symmetric and between-field synaptic values are symmetrizable. Our discussion focuses more on encoding than on decoding properties of competitive learning systems. So the discussion is limited to feedforward autoassociative competitive learning systems for estimating unknown probability density functions $p(\mathbf{x})$. Heteroassociative extensions are immediate.

Competitive Learning as Adaptive Vector Quantization

Competitive learning systems adaptively quantize the pattern space R^n. The synaptic vector $\mathbf{m}_j$ represents, or "rounds off," the local region about $\mathbf{m}_j$. Each synaptic vector $\mathbf{m}_j$ is a *quantization vector*. The competitive learning system learns as synaptic vectors $\mathbf{m}_j$ change in response to randomly sampled training data. Geometrically, learning occurs if and only if the synaptic vectors $\mathbf{m}_j$ move in the pattern space R^n.

Competitive learning adaptively distributes the m synaptic vectors $\mathbf{m}_1, \ldots, \mathbf{m}_m$ in R^n to approximate the *unknown* probability density function $p(\mathbf{x})$ of the random pattern vector $\mathbf{x}$. The patterns $\mathbf{x}$ are continuously distributed in R^n. $p(\mathbf{x})$ describes their distribution. Where the patterns $\mathbf{x}$ are dense or sparse, the synaptic vectors $\mathbf{m}_j$ tend to be dense or sparse. Different competitive learning, or *adaptive vector quantization* (AVQ), schemes distribute the synaptic vectors in different ways.

If $p(\mathbf{x})$ were known, learning would be unnecessary. Numerical techniques[3,22] could directly determine pattern clusters or classes, centroids, and class boundaries.

All observed patterns are realizations of a single *random vector* $\mathbf{x}$. The random vector $\mathbf{x}$ can be interpreted as n ordered scalar random variables: $\mathbf{x} = (x_1, \ldots, x_n)$. This is heuristic but incomplete.

A random vector $\mathbf{x}$ is a function. It is a *measurable* function from a sample space to a vector space. (This means[19] inverse images $\mathbf{x}^{-1}(O)$ of measurable or open subsets O of the vector space are measurable subsets of the sample space.) In the autoassociative case both spaces are R^n, so $\mathbf{x} : R^n \rightarrow R^n$. The sigma algebra of measurable sets is the Borel[1] sigma algebra $B(R^n)$, the topological sigma algebra generated[19] from the open subsets of R^n. In practice the random vector $\mathbf{x} : R^n \rightarrow R^n$ is just the identity function: $\mathbf{x}(\mathbf{v}) = \mathbf{v}$ for all $\mathbf{v}$ in R^n.

The *cumulative distribution function* $P : B(R^n) \rightarrow [0, 1]$ maps open subsets of R^n to numbers in $[0, 1]$ and is countably additive on countably-infinite disjoint unions of subsets of R^n. $p(\mathbf{x})$ characterizes the "randomness" of the random vector $\mathbf{x}$. $P(O)$ is the integral of $p(\mathbf{x})$ on the open set $O \subset R^n$. Since $\mathbf{x}$ is the identity function on R^n, $p(\mathbf{x})$ characterizes the *occurrence probability* of the observed pattern samples or realizations used in training or recognition.

The stochastic pattern recognition framework is rigorously defined by specifying the probability space $(R^n, B(R^n), P)$ and the pattern random-vector function $\mathbf{x}$. In the default case $\mathbf{x}$ is the identity function.

Pattern clusters or classes are subsets of R^n. Some pattern classes are more probable than others. The pattern space R^n is partitioned into k subsets or *decision classes* $D_1, \ldots, D_k$:

$$R^n = D_1 \cup \ldots \cup D_k \text{ and } D_i \cap D_j = \emptyset \text{ if } i \neq j \quad . \tag{1}$$

The distinction between supervised and unsupervised pattern learning and recognition depends on the available information. In both cases the probability density function $p(\mathbf{x})$ is unknown. That is why adaptive techniques are used instead of, say, numerical optimization or calculus-of-variation[3] techniques.

Supervised learning requires more information than unsupervised learning. Unsupervised learning uses minimal information. Pattern learning is *supervised* if the decision classes $D_1, \ldots, D_k$ are known and the learning system uses this information. The user knows—and the algorithm uses—the class membership of every sample pattern $\mathbf{x}$. The user knows that $\mathbf{x} \in D_i$ and that $\mathbf{x} \notin D_j$ for all $j \neq i$. Learning is *unsupervised* if class memberships are unknown. Supervised learning systems allow an error measure or vector to be computed. The simplest error measure is the desired outcome minus the actual outcome. The error measure guides the learning process with feedback error correction.

AVQ Class Probability Estimation

The partition property (1) implies that $p(D_1)+\ldots+p(D_k) = 1$ since $p(R^n) = 1$. The *class probability* $p(D_i)$ is given by

$$p(D_i) = \int_{D_i} p(\mathbf{x})\, d\mathbf{x} \tag{2}$$

$$= E[I_{D_i}] \quad , \tag{3}$$

where the integral in (2) is an n-dimensional multiple integral. $E[x]$ is the expectation of random variable x. The function $I_S : R^n \rightarrow \{0, 1\}$ is the *indicator* function of set S. $I_S(\mathbf{x}) = 1$ if $\mathbf{x} \in S$, $I_S(\mathbf{x}) = 0$ if $\mathbf{x} \notin S$. In the probabilistic setting the indicator function I_S is random (Borel measurable[1]), and hence a random variable.

A pattern $\mathbf{x}$ is in exactly one decision class—with probability one. With probability zero, pattern $\mathbf{x}$ can be on the border of two or more decision classes. $p(\mathbf{x}) = 0$ for every $\mathbf{x}$ in R^n.

A *uniform partition* gives $p(D_i) = 1/k$ for each decision class D_i in the partition. Uniform partitions are clearly not unique. Some vector quantization schemes attempt to adaptively partition R^n into a uniform partition. Then it should be equally likely that a pattern sample $\mathbf{x}$ drawn at random (according to $p(\mathbf{x})$) from R^n was drawn from any one of the k decision classes D_i. This corresponds to each competing neuron "winning" with the same frequency. Competitive learning has been modified[2,20] in several, usually supervised, ways to force the competing neurons to have the same win rate. The motivation for such modifications is economy: fewer neurons are needed to estimate a sampled continuous function.

Nonuniform partitions are more informative than uniform partitions. They also occur more frequently when estimating an unknown probability density function $p(\mathbf{x})$. When the number of competing neurons is less than the number of distinct pattern classes, when $m < k$, some neurons win more frequently than others. If $p(D_i) > p(D_j)$, the competing neuron that codes for D_i tends to win more frequently than the neuron that codes for D_j. Equivalently, more sample patterns $\mathbf{x}$ tend to be closer in Euclidean distance to the corresponding synaptic vector, call it $\mathbf{m}_i$, that quantizes D_i than to the synaptic vector $\mathbf{m}_j$ that quantizes D_j. Below we show that $\mathbf{m}_i$ and $\mathbf{m}_j$ tend to arrive at the respective centroids of D_i and D_j. Centroids minimize the mean-squared error of vector quantization[18].

In general there are more competing neurons than decision classes, $m > k$. For neurons can always be added to the competitive learning system. Then if $p(D_i) > p(D_j)$, there tend to be more synaptic vectors within D_i than within D_j. In principle all the neurons corresponding to the synaptic vectors in D_i can have the same win rates. But since metrical classification is used to decide which neuron wins, neurons with synaptic vectors nearer the centroid of D_i tend to win more frequently.

The number of synaptic vectors in decision class D_i gives a nonparametric estimate of the class probability $p(D_i)$: $p(D_i) = \frac{n_i}{m}$, where n_i is the number of synaptic vectors in D_i. In general the quantizing synaptic vectors nonparametrically estimate the probability density function $p(\mathbf{x})$. No probability assumptions need be made about the observed training samples. For any subset or volume $V \subset R^n$, the volume probability $p(V)$ is estimated as the ratio

$$p(V) = \frac{n_V}{m} \quad , \tag{4}$$

where n_V is the number of synaptic vectors $\mathbf{m}_j$ in V and m is the total number of synaptic vectors. In the extreme case (4) gives $p(R^n) = 1$ and $p(\emptyset) = 0$.

Deterministic Competitive Learning Laws

The idea behind competitive learning is *learn only if win*. Losing neurons, or rather their synaptic fan-in vectors, do not learn. They also do not forget what they have already learned. The price is a nondistributed representation[5] The synapses in a synaptic vector $\mathbf{m}_j$ become in effect "grandmother" synapses. Each synaptic elemen. m_{ij} is a discrete memory unit, as in a random access memory.

In contrast, classical Hebbian[6] or correlation learning distributes learned pattern-vector information across the entire synaptic connection matrix M. But a Hebbian system forgets learned pattern information as it learns new pattern information.

The simplest deterministic *competitive learning*[5,16,20] law is, in component-wise notation,

$$\dot{m}_{ij} = S_j(y_j)[S_i(x_i) - m_{ij}] \quad , \tag{5}$$

where $\dot{m}_{ij}$ is the time derivative of the synaptic value of the directed axonal connection from the ith neuron in the input field F_X to the jth neuron in the output or competitive field F_Y. The n-by-m matrix M consists of the m_{ij} values. The jth column of M is the fan-in synaptic vector $\mathbf{m}_j = (m_{1j}, \ldots, m_{nj})$. Scaling constants can be added or multiplied in (5) as desired. In contrast, the *signal Hebbian learning*[5,14] law is

$$\dot{m}_{ij} = -m_{ij} + S_i(x_i)S_j(y_j) \quad . \tag{6}$$

(5) and (6) differ in how they forget. All learning requires some forgetting. The competitive signal S_j in (5) nonlinearly scales the decay or forget term $-m_{ij}$. In practice[7,11-12,20] the competitive signal S_j is a zero-one or binary threshold function. Winners forget, losers remember.

There are n neurons in F_X and m competing neurons in F_Y. Each neuron in F_X or F_Y is a function that transduces its real-valued *activation* $x_i(t)$ or $y_j(t)$ into a bounded *signal* $S_i(x_i(t))$ or $S_j(y_j(t))$ at time t. In principle the activation functions, or membrane potential differences, x_i and y_j can be unbounded.

In feedback networks, the signal functions S_i and S_j are usually assumed bounded and monotone nondecreasing. So their activation derivatives S_i' and S_j' are nonnegative. In practice logistic or hyperbolic-tangent signal functions are often used. Then the signal functions S_i and S_j are strictly increasing and hence their activation derivatives are positive:

$$S_i' = \frac{dS_i}{dx_i} > 0 \quad \text{and} \quad S_j' = \frac{dS_j}{dy_j} > 0 \quad . \tag{7}$$

For instance, the logistic signal function $S(x) = (1 - e^{-cx})^{-1}$ with scale constant $c > 0$ has an increasing activation derivative $S' = c\,S\,(1 - S) > 0$. The logistic signal function rapidly approaches a binary threshold function for increasing values of c.

In competitive learning the F_Y signal functions S_j are often binary threshold functions. $S_j(t) = 1$ if the jth competing neuron in F_Y wins the competition for activation at time t. $S_j(t) = 0$ if the jth neuron loses.

The F_X signal functions S_i are usually linear in feedforward systems: $S_i(x_i) = x_i$. Then the sample pattern $\mathbf{x} = (x_1, \ldots, x_n)$ directly activates the system as the F_X signal state vector $S_X(\mathbf{x}) = \mathbf{x}$. So in practice the competitive learning law (5) becomes

$$\dot{m}_{ij} = I_{D_j}(\mathbf{x})\,[x_i - m_{ij}] \quad , \tag{8}$$

where I_{D_j} is the zero-one indicator function of decision class D_j. We assume synaptic vector $\mathbf{m}_j$ codes for class D_j, perhaps by hovering about the centroid of D_j.

Kohonen's recent[11] *supervised competitive learning* (SCL) law is a reinforced version of (5):

$$\dot{m}_{ij} = r_j(\mathbf{x})\,S_j\,[x_i - m_{ij}] \quad , \tag{9}$$

where S_j is a binary threshold function determined metrically. $S_j = 1$ if $\mathbf{x}$ is closer in Euclidean distance to the synaptic vector $\mathbf{m}_j$ than to all other synaptic vectors $\mathbf{m}_i$. The new term r_j in (8) is the *reinforcement function* of the jth competing neuron in F_Y. r_j rewards when $r_j(\mathbf{x}) = 1$ and punishes when $r_j(\mathbf{x}) = -1$.

The reinforcement function r_j is determined by the class membership of the pattern sample $\mathbf{x}$. So (9) is a supervised competitive learning law. $r_j(\mathbf{x}) = 1$ if $\mathbf{x} \in D_j$ and the jth neuron wins or correctly "classifies" $\mathbf{x}$—if $I_{D_j}(\mathbf{x}) = S_j(\mathbf{x}) = 1$. $r_j(\mathbf{x}) = -1$ if the winning jth neuron misclassifies the sample pattern $\mathbf{x}$. Misclassification means the jth neuron wins but $\mathbf{x} \in D_i$, or $\mathbf{x} \notin D_j$, for some $i \neq j$. Then $I_{D_j}(\mathbf{x}) = 0$ but $I_{D_i}(\mathbf{x}) = S_j = 1$. Since, with probability one, $\mathbf{x}$ belongs to exactly one decision class, the reinforcement function reduces to a difference of decision-class indicator functions:

$$r_j = I_{D_j} - \sum_{i \neq j} I_{D_i} \quad . \tag{10}$$

(10) makes explicit the dependence of r_j on the knowledge of the decision class boundaries.

The unsupervised *differential competitive learning*[16] (DCL) law modulates the vector difference $\mathbf{x} - \mathbf{m}_j$ with the competitive *win rate* $\dot{S}_j$:

$$\dot{m}_{ij} = \dot{S}_j(y_j)\,[S_i(x_i) - m_{ij}] \quad , \tag{11}$$

where the signal velocity $\dot{S}_j$ decomposes as $S_j' \dot{y}_j$ by the chain rule. The idea is *learn only if change*. The signal velocity in (11) behaves in sign much as the reinforcement function in (9). The signal velocity $\dot{S}_j(t)$ is positive or negative according as the jth competing neuron's winning status is increasing or decreasing at time t. The signal velocity does not depend on the decision-class indicator functions. So the DCL law (11) is unsupervised.

In practice the F_X signal function S_i is linear. Then simulations[12] show that the DCL law and Kohonen's SCL law (9) behave similarly. The DCL systems tend to converge to decision class centroids at least as fast as SCL systems do and tend to wander about the centroids with less variance. The competitive learning laws (5) and (9) ignore the win-rate information provided by the signal velocity in (11).

The *pulse-coded*[4,16] signal function S_j is an exponentially weighted average of binary pulses:

$$S_j(t) = \int_{-\infty}^{t} y_j(s)\, e^{s-t}\, ds \quad , \tag{12}$$

where the pulse function y_j is defined by $y_j(t) = 1$ if a pulse is present at time t and $y_j(t) = 0$ if no pulse is present. Then the signal velocity is the simple, locally available, difference

$$\dot{S}_j(t) = y_j(t) - S_j(t) \quad . \tag{13}$$

The velocity-difference representation (13) eliminates the need for an approximation algorithm to calculate the signal velocity. Biological, or silicon, synapses can modify their values in realtime with signal velocity information. Biological neurons transmit and receive pulse trains, not real-valued sigmoidal outputs. The presence or absence of a pulse is easier to detect, amplify, and emit than a multi-valued signal. (13) shows that much of the time the arriving pulse $y_j(t)$ indicates the instantaneous sign of the signal velocity.

The pulse-coded differential competitive law approximates[16] the classical competitive law (5) as can be seen by substituting (13) into (11) and expanding terms. A related approximation of the signal Hebb law (6) occurs when (13) eliminates a product of signal velocities in a comparable *differential Hebbian learning*[9-10,13,15-16] law.

Stochastic Competitive Learning Laws and Algorithms

Stochastic competitive learning laws are stochastic differential equations. They describe how synaptic random processes change as a function of other random processes. Their solution is a synaptic random process[21].

The deterministic competitive learning laws (5), (9), and (11) are simple stochastic differential equations if the signal terms $S_i(x_i(t))$ are random variables at each time t. This is so when the sample vectors $\mathbf{x}$ are random samples, realizations of the pattern random-vector process $\mathbf{x}: R^n \rightarrow R^n$. The randomness in the vector components x_i induces randomness in the signal function S_i and thus in the synaptic vectors $\mathbf{m}_j$. In general each term in a stochastic differential equation is a random process.

Another simple stochastic differential equation arises when random noise is added to a differential equation. The randomness in the noise process induces randomness in the dependent variables. In general, and in this discussion, an independent noise process is added to a stochastic differential equation.

The stochastic competitive learning law is, in vector notation,

$$\mathbf{dm}_j = S_j(y_j)\,[\mathbf{S(x)} - \mathbf{m}_j]\,dt + \mathbf{dB}_j \quad , \tag{14}$$

where S_j is a steep competitive signal process taking values in $[0, 1]$ and $\mathbf{S(x)} = (S_1(x_1), \ldots, S_n(x_n))$ for random pattern $\mathbf{x}$. $\mathbf{B}_j$ is a Brownian motion diffusion process.

The pseudo-derivative[21] of $\mathbf{B}_j$ is the zero-mean *white noise* process $\mathbf{n}_j$. The pseudo-derivative can be formed as a mean-squared limit. The noise process $\mathbf{n}_j$ is zero-mean, $E[n_{ij}] = 0$, has finite variance, and is independent of the "signal" term $S_j(y_j)\,[\mathbf{S(x)} - \mathbf{m}_j]$. Then competitive learning laws can be written in less rigorous, more intuitive, "noise" notation. For example, (14) becomes

$$\dot{\mathbf{m}}_j = S_j(y_j)\,[\mathbf{S(x)} - \mathbf{m}_j] + \mathbf{n}_j \quad . \tag{15}$$

In practice S_j is a binary threshold function and can often be replaced with the class indicator function I_{D_j}. The F_X signal processes S_i are linear. So (15) becomes

$$\dot{\mathbf{m}}_j = I_{D_j}(\mathbf{x})[\mathbf{x} - \mathbf{m}_j] + \mathbf{n}_j \quad . \tag{16}$$

The stochastic version of Kohonen's supervised competitive learning (SCL) law is

$$\dot{\mathbf{m}}_j = r_j(\mathbf{x})\,S_j(y_j)\,[\mathbf{x} - \mathbf{m}_j] + \mathbf{n}_j \quad . \tag{17}$$

The stochastic version of the differential competitive learning (DCL) law is

$$\dot{\mathbf{m}}_j = \dot{S}_j(y_j)[\mathbf{x} - \mathbf{m}_j] + \mathbf{n}_j \quad , \tag{18}$$

or, in pulse-coded form,

$$\dot{\mathbf{m}}_j = [y_j(t) - S_j(t)]\,[\mathbf{x} - \mathbf{m}_j] + \mathbf{n}_j \ , \tag{19}$$

where the pulse process y_j is a random *point process*, perhaps Poisson in nature. (19) is thus a doubly stochastic synaptic model.

For practical implementation these three stochastic competitive learning models can be written as stochastic difference equations by replacing the third step in the following competitive AVQ algorithm. (Historical note: Tsypkin[22] derived the "winning" parts of the UCL algorithm and, with his adaptive Bayes approach, the SCL algorithm in a non-neural context.) A random noise term has not been added to the difference equations. The noise processes in the above models can represent unmodeled effects, roundoff errors, or sample-size defects.

Competitive AVQ Algorithms

1. Initialize synaptic vectors: $\mathbf{m}_i(0) = \mathbf{x}(i),\ i = 1, \ldots, m$.
2. For random sample $\mathbf{x}(t)$, find the closest ("winning") synaptic vector $\mathbf{m}_j(t)$:

$$\|\mathbf{m}_j(t) - \mathbf{x}(t)\| = \min_i \|\mathbf{m}_i(t) - \mathbf{x}(t)\| \ , \tag{20}$$

   where $\|\mathbf{x}\|^2 = x_1^2 + \ldots + x_n^2$ is the squared Euclidean norm of $\mathbf{x}$.
3. Update the winning synaptic vector(s) $\mathbf{m}_j(t)$ by the UCL, SCL, or DCL learning algorithm.

Unsupervised Competitive Learning (UCL)

$$\begin{aligned} \mathbf{m}_j(t+1) &= \mathbf{m}_j(t) + c_t[\mathbf{x}(t) - \mathbf{m}_j(t)] \ , \\ \mathbf{m}_i(t+1) &= \mathbf{m}_i(t) \quad \text{if} \quad i \neq j \ , \end{aligned} \tag{21}$$

where, in the spirit of stochastic approximation[22], c_t is a slowly decreasing sequence of learning coefficients. For instance, $c_t = .1(1 - \frac{t}{10{,}000})$ for 10,000 samples $\mathbf{x}(t)$.

Supervised Competitive Learning (SCL)

$$\mathbf{m}_j(t+1) = \mathbf{m}_j(t) + c_t\, r_j(\mathbf{x}(t))\,[\mathbf{x}(t) - \mathbf{m}_j(t)] \tag{22}$$

$$= \begin{cases} \mathbf{m}_j(t) + c_t\,[\mathbf{x}(t) - \mathbf{m}_j(t)] & \text{if } \mathbf{x} \text{ correctly classified} \\ \mathbf{m}_j(t) - c_t\,[\mathbf{x}(t) - \mathbf{m}_j(t)] & \text{if } \mathbf{x} \text{ misclassified.} \end{cases} \tag{23}$$

Differential Competitive Learning (DCL)

$$\begin{aligned} \mathbf{m}_j(t+1) &= \mathbf{m}_j(t) + c_t \Delta S_j(y_j(t))\,[\mathbf{x}(t) - \mathbf{m}_j(t)] \ , \\ \mathbf{m}_i(t+1) &= \mathbf{m}_i(t) \ \text{ if } \ i \neq j \ , \end{aligned} \tag{24}$$

where $\Delta S_j(y_j(t))$ is the time change of the jth neuron's competitive signal $S_j(y_j)$ in the competition field F_Y:

$$\Delta S_j(y_j(t)) = S_j(y_j(t+1)) - S_j(y_j(t)) \quad . \tag{25}$$

In practice[12] only the sign of the difference (25) may be used. The F_Y neuronal activations y_j can be updated by an *additive* model:

$$y_j(t+1) = y_j(t) + \sum_{i=1}^{n} S_i(x_i)\, m_{ij}(t) + \sum_{k=1}^{m} S_k(y_k)\, w_{kj} \quad . \tag{26}$$

The fixed competition matrix W defines a symmetric *lateral inhibition* topology within F_Y. In the simplest case, $w_{jj} = 1$ and $w_{ij} = -1$ for distinct i and j.

Stochastic Equilibrium and Convergence

Competitive synaptic vectors $\mathbf{m}_j$ converge to decision class centroids. The centroids may be local maxima of the sampled but unknown probability density function $p(\mathbf{x})$.

In general, when there are more synaptic vectors than probability maxima, the synaptic vectors cluster about local probability maxima. Comparatively few synaptic vectors may actually arrive at centroids. We only consider convergence to centroids. The justification is that any local connected patch of the sample space R^n can be viewed as a candidate decision class. Each synaptic vector samples such a local patch and converges to its centroid.

We first prove the AVQ Centroid Theorem: If a competitive AVQ system converges, it converges to the centroid of the sampled decision class. The AVQ Centroid Theorem is an equilibrium or steady-state result. We prove the theorem only for unsupervised competitive learning, but argue that it holds for supervised and differential competitive learning in most cases of practical interest.

Next we use a Lyapunov argument to reprove and extend the AVQ Centroid Theorem to the AVQ Convergence Theorem: Stochastic competitive learning systems are asymptotically stable, and synaptic vectors converge to centroids. So competitive AVQ systems always converge, and converge exponentially fast. Both results are true with probability one.

The unknown probability density function $p(\mathbf{x})$ defines the class centroids, the mean-squared optimal vectors of quantization. Competitive learning estimates these optimal quantization vectors without knowledge of $p(\mathbf{x})$. That is the advantage of competitive learning, and optimal learning in general.

AVQ Centroid Theorem:

$$\text{Prob}(\mathbf{m}_j = \bar{\mathbf{x}}_j) = 1 \quad \text{at equilibrium} \; . \tag{27}$$

The centroid $\bar{\mathbf{x}}_j$ of decision class D_j is defined by

$$\bar{\mathbf{x}}_j = \frac{\int_{D_j} \mathbf{x}\, p(\mathbf{x})\, d\mathbf{x}}{\int_{D_j} p(\mathbf{x})\, d\mathbf{x}} \tag{28}$$

$$= E[\mathbf{x} | \mathbf{x} \in D_j] \quad . \tag{29}$$

The random vector $E[\mathbf{x}|\cdot]$, the conditional expectation, is a function of Borel measurable subsets D_j of R^n.

Proof. Suppose the jth neuron in F_Y wins the activation competition during the training interval. Suppose the jth synaptic vector $\mathbf{m}_j$ codes for decision class D_j. So $I_{D_j}(\mathbf{x}) = 1$ iff $S_j = 1$. Suppose stochastic equilibrium has been reached:

$$\dot{\mathbf{m}}_j = \mathbf{O} \quad , \tag{30}$$

which holds with probability one (or in the mean-square sense, depending on how the stochastic differentials are defined). Take expectations of both sides of (30), use the zero-mean property of the noise process, eliminate the synaptic velocity vector $\dot{\mathbf{m}}_j$ with the competitive law (16), and expand to give

$$\mathbf{O} = E[\dot{\mathbf{m}}_j] \tag{31}$$

$$= \int_{R^n} I_{D_j}(\mathbf{x})\,(\mathbf{x} - \mathbf{m}_j)\,p(\mathbf{x})\,d\mathbf{x} + E[\mathbf{n}_j] \tag{32}$$

$$= \int_{D_j} (\mathbf{x} - \mathbf{m}_j)\,p(\mathbf{x})\,d\mathbf{x} \tag{33}$$

$$= \int_{D_j} \mathbf{x}\,p(\mathbf{x})\,d\mathbf{x} - \mathbf{m}_j \int_{D_j} p(\mathbf{x})\,d\mathbf{x} \quad , \tag{34}$$

since $\mathbf{m}_j$ is constant with probability one. Solving for the equilibrium synaptic vector $\mathbf{m}_j$ gives the centroid $\bar{\mathbf{x}}_j$ defined by (28). Q.E.D.

In general the AVQ Centroid Theorem concludes that the *average* synaptic vector $E[\mathbf{m}_j]$ equals the jth centroid $\bar{\mathbf{x}}_j$ at equilibrium:

$$E[\mathbf{m}_j] = \bar{\mathbf{x}}_j \quad . \tag{35}$$

The equilibrium synaptic vector $\mathbf{m}_j$ vibrates randomly around the constant centroid $\bar{\mathbf{x}}_j$. $\mathbf{m}_j$ equals $\bar{\mathbf{x}}_j$ on average at each post-equilibration instant. Simulations[12] exhibit such random wandering about the centroid.

Synaptic vectors learn noise as well as signal. So they vibrate at equilibrium. The independent additive noise process $\mathbf{n}_j$ in (16) drives the random vibration. The steady-state condition (30) models the rare event that noise cancels signal. In general it models stochastic equilibrium in the absence of additive noise.

The general *stochastic steady-state condition* is defined by the stochastic differential equation

$$\dot{\mathbf{m}}_j = \mathbf{n}_j \quad . \tag{36}$$

Taking expectations of both sides of (36) still gives (31), since the noise process $\mathbf{n}_j$ is zero-mean, and the argument proceeds as before. Taking a second expectation in (33) and using (31) gives (35).

The AVQ Centroid Theorem applies to the stochastic SCL law (17) because winners are picked metrically by the nearest-neighbor criterion (20). The reinforcement function r_j in (10) reduces to $r_j(\mathbf{x}) = -I_{D_i}(\mathbf{x}) = -1$ if the jth neuron continually wins for random samples $\mathbf{x}$ from class D_i. This tends to occur once the synaptic vectors have spread out in R^n and D_i is close, usually contiguous, to D_j. Then $\mathbf{m}_j$ converges to $\bar{\mathbf{x}}_i$, the centroid of D_i, since the steady state condition (30) removes the scaling constant -1 that then appears in (33).

This argument holds only approximately when, in the exceptional case, $\mathbf{m}_j$ repeatedly misclassifies patterns $\mathbf{x}$ from several classes D_k. Then the difference of indicator functions in (10) replaces the single

indicator function I_{D_j} in (32). The resultant equilibrium $\mathbf{m}_j$ is a more general ratio than the centroid. The density $p(\mathbf{x})$ must be integrated over R^n not just D_j.

The AVQ Centroid Theorem applies similarly to the stochastic DCL law (18). A positive or negative factor scales the difference $\mathbf{x} - \mathbf{m}_j$. If, as in practice and in (24), a constant approximates the scaling factor, the steady state condition (30) removes the constant from (33) and $\mathbf{m}_j$ estimates the centroid $\bar{\mathbf{x}}_j$.

The integrals in (31) - (34) are *spatial* integrals over R^n or subsets of R^n. Yet in the discrete UCL, SCL, and DCL algorithms, the recursive equations for $\mathbf{m}_j(t+1)$ define *temporal* integrals over the training interval.

The two integrals are approximately equal. The discrete random samples $\mathbf{x}(0), \mathbf{x}(1), \mathbf{x}(2), \ldots$ partially enumerate the continuous distribution of equilibrium realizations of the random vector $\mathbf{x}$. The time index in the discrete algorithms approximates the "spatial index" underlying $p(\mathbf{x})$. So the recursion $\mathbf{m}_j(t+1) = \mathbf{m}_j(t) + \ldots$ approximates the averaging integral. We sample patterns one at a time. We integrate them all at a time.

The AVQ Centroid Theorem assumes that stochastic convergence occurs. Convergence is trivial for continuous deterministic competitive learning, at least in feedforward networks. If S_j is a positive constant in (5), then m_{ij} converges to S_i exponentially fast. Convergence is not trivial for stochastic competitive learning in noise.

The AVQ Convergence Theorem ensures exponential convergence. The theorem does not depend on how the F_Y neurons change in time. In effect metrical classification is assumed: $S_j = 1$ iff $I_{D_j}(\mathbf{x}) = 1$. The strictly decreasing deterministic Lyapunov function $E[L]$ replaces[16] the random Lyapunov function $L: C \rightarrow R$, where C is a closed and bounded (compact) subset of R^m.

A strictly decreasing Lyapunov function yields asymptotic stability[17]. Then the real parts of the eigenvalues of the system Jacobian matrix are strictly negative, and locally the nonlinear system behaves linearly. Synaptic vectors converge[8] exponentially quickly to equilibrium points—to pattern-class centroids—in the state space. Technically, nondegenerate Hessian matrix conditions must be assumed. Else some eigenvalues can have zero real parts.

AVQ Convergence Theorem: ***Competitive synaptic vectors converge exponentially quickly to pattern-class centroids.***

Proof. Consider the random quadratic form L:

$$L = \frac{1}{2} \sum_{i}^{n} \sum_{j}^{m} (x_i - m_{ij})^2 \quad . \tag{37}$$

Note that if $\mathbf{x} = \bar{\mathbf{x}}_j$ in (37), then with probability one $L > 0$ if any $\mathbf{m}_j \neq \bar{\mathbf{x}}_j$ and $L = 0$ iff $\mathbf{m}_j = \bar{\mathbf{x}}_j$ for every $\mathbf{m}_j$.

The pattern vectors $\mathbf{x}$ do not change in time. (The following argument is still valid if the pattern vectors $\mathbf{x}$ change slowly relative to synaptic changes—if the density $p(\mathbf{x})$ is mildly nonstationary.) This simplifies the stochastic derivative of L:

$$\dot{L} = \sum_{i} \frac{\partial L}{\partial x_i} \dot{x}_i + \sum_{i} \sum_{j} \frac{\partial L}{\partial m_{ij}} \dot{m}_{ij} \tag{38}$$

$$= \sum_{i} \sum_{j} \frac{\partial L}{\partial m_{ij}} \dot{m}_{ij} \tag{39}$$

$$= - \sum_i \sum_j (x_i - m_{ij})\, \dot{m}_{ij} \tag{40}$$

$$= - \sum_i \sum_j I_{D_j}(\mathbf{x})\, (x_i - m_{ij})^2 - \sum_i \sum_j (x_i - m_{ij})\, n_{ij} \quad . \tag{41}$$

L is a random variable at every time t. $E[L]$ is a deterministic number at every t. The trick is to use the average $E[L]$ as a Lyapunov function for the stochastic competitive dynamical system. For this we must assume sufficient smoothness to interchange the time derivative and the probabilistic integral—to bring the time derivative "inside" the integral. Then the zero-mean noise assumption, and the independence of the noise process $\mathbf{n}_j$ with the "signal" process $\mathbf{x} - \mathbf{m}_j$, gives

$$\dot{E}[L] = E[\dot{L}] \tag{42}$$

$$= - \sum_j \int_{D_j} \sum_i (x_i - m_{ij})^2\, p(\mathbf{x})\, d\mathbf{x} \quad . \tag{43}$$

So, on average by the learning law (16), $\dot{E}[L] < 0$ iff any synaptic vector $\mathbf{m}_j$ moves along its trajectory. So the competitive AVQ system is *asymptotically* stable[8,17] and, in general, converges exponentially quickly to equilibria.

Suppose $\dot{E}[L] = 0$. Every synaptic vector has reached equilibrium and is constant (with probability one). Then[19], since $p(\mathbf{x})$ is a nonnegative weight function, the weighted integral of the learning differences $x_i - m_{ij}$ must also be zero:

$$\int_{D_j} (\mathbf{x} - \mathbf{m}_j)\, p(\mathbf{x})\, d\mathbf{x} = \mathbf{0} \tag{44}$$

in vector notation. (44) is identical to (33). So, with probability one, equilibrium synaptic vectors are centroids. More generally, as discussed above, (35) holds. Average equilibrium synaptic vectors are centroids: $E[\mathbf{m}_j] = \bar{\mathbf{x}}_j$. Q.E.D.

The sum of integrals (43) defines the total mean-squared error of vector quantization for the partition $D_1, \ldots, D_k$. The vector integral in (44) is the gradient of $\dot{E}[L]$ with respect to $\mathbf{m}_j$. So the AVQ Convergence Theorem implies that class centroids—and, asymptotically, competitive synaptic vectors—minimize the mean-squared error of vector quantization.

Then by (16), the synaptic vectors perform stochastic gradient descent on the mean-squared-error surface in the pattern-plus-error space R^{n+1}. The difference $\mathbf{x}(t) - \mathbf{m}_j(t)$ behaves as an error vector. The competitive system estimates the unknown centroid $\bar{\mathbf{x}}_j$ as $\mathbf{x}(t)$ at each time t. Learning is unsupervised but proceeds *as if* it were supervised.

Acknowledgment

This research was supported by the Air Force Office of Scientific Research (AFOSR-88-0236) and by a grant from the Shell Oil Corporation.

References

[1] Chung, K.L, *A Course in Probability Theory*, New York: Academic Press, 1974.

[2] DeSieno, D., "Adding a Conscience to Competitive Learning," *Proceedings of the 2nd IEEE International Conference on Neural Networks (ICNN-88)*, vol. I, 117 - 124, July 1988.

[3] Fleming, W.H, Rishel, R.W., *Deterministic and Stochastic Optimal Control*, Springer-Verlag, 1975.

[4] Gluck, M.A., Parker, D.B., Reifsnider, E., "Some Biological Implications of a Differential-Hebbian Learning Rule," *Psychobiology*, vol. 16, no. 3, 298 - 302, 1988.

[5] Grossberg, S., "On Learning and Energy-Entropy Dependence in Recurrent and Nonrecurrent Signed Networks," *Journal of Statistical Physics*, vol. 1, 319 - 350, 1969.

[6] Hebb, D.O, *The Organization of Behavior*, Wiley, 1949.

[7] Hecht-Nielsen, R., "Counterpropagation Networks," *Applied Optics*, vol. 26, no. 3, 4979 -4984, 1 December 1987.

[8] Hirsch, M.W., Smale, S., *Differential Equations, Dynamical Systems, and Linear Algebra*, New York: Academic Press, 1974.

[9] Klopf, A.H., "A Drive-Reinforcement Model of Single Neuron Function: An Alternative to the Hebbian Neuronal Network," *Proceedings American Institute of Physics: Neural Networks for Computing*, 265 - 270, April 1986.

[10] Klopf, A.H., "A Neuronal Model of Classical Conditioning," *Psychobiology*, vol. 16, no. 2, 85 -125, 1988.

[11] Kohonen, T., *Self-Organization and Associative Memory*, Second Edition, Springer-Verlag, 1988.

[12] Kong, S.G., Kosko, B., "Differential Competitive Learning for Centroid Estimation and Phoneme Recognition," submitted for publication, February 1990.

[13] Kosko, B., "Differential Hebbian Learning," *Proceedings American Institute of Physics: Neural Networks for Computing*, 277 -282, April 1986.

[14] Kosko, B., "Adaptive Bidirectional Associative Memories," *Applied Optics*, vol. 26, no. 23, 4947 - 4960, 1 December 1987.

[15] Kosko, B., "Hidden Patterns in Combined and Adaptive Knowledge Networks," *International Journal of Approximate Reasoning*, vol. 2, no. 4, 377 - 393, October 1988.

[16] Kosko, B., "Unsupervised Learning in Noise," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, 44 - 57,March 1990.

[17] Parker, T.S., Chua, L.O., "Chaos: A Tutorial For Engineers," *Proceedings of the IEEE*, vol. 75, no. 8, 982 -1008, August 1987.

[18] Pratt, W.K., *Digital Image Processing*, Wiley: New York, 1978.

[19] Rudin, W., *Real and Complex Analysis*, Second Edition, McGraw-Hill, 1974.

[20] Rumelhart, D.E., Zipser, D., "Feature Discovery by Competitive Learning," *Cognitive Science*, vol. 9, 75 - 112, 1985.

[21] Skorokhod, A.V., *Studies in the Theory of Random Processes*, Reading, MA: Addison-Wesley, 1965.

[22] Tsypkin, Ya. Z., *Foundations of the Theory of Learning Systems*, Academic Press, 1973.